```
RRRRRRRRRRR      UUU          UUU  NNN         NNN     000000000     FFFFFFFFFFFFFFF  FFFFFFFFFFFFFFF
RRRRRRRRRRRR     UUU          UUU  NNN         NNN     000000000     FFFFFFFFFFFFFFF  FFFFFFFFFFFFFFF
RRRRRRRRRRRR     UUU          UUU  NNN         NNN     000000000     FFFFFFFFFFFFFFF  FFFFFFFFFFFFFFF
RRR          RRR UUU          UUU  NNN         NNN  000         000  FFF              FFF
RRR          RRR UUU          UUU  NNN         NNN  000         000  FFF              FFF
RRR          RRR UUU          UUU  NNN         NNN  000         000  FFF              FFF
RRR          RRR UUU          UUU  NNNNNN      NNN  000         000  FFF              FFF
RRR          RRR UUU          UUU  NNNNNN      NNN  000         000  FFF              FFF
RRR          RRR UUU          UUU  NNNNNN      NNN  000         000  FFF              FFF
RRRRRRRRRRRR     UUU          UUU  NNN  NNN    NNN  000         000  FFFFFFFFFFFF     FFFFFFFFFFFF
RRRRRRRRRRRR     UUU          UUU  NNN  NNN    NNN  000         000  FFFFFFFFFFFF     FFFFFFFFFFFF
RRRRRRRRRRRR     UUU          UUU  NNN  NNN    NNN  000         000  FFFFFFFFFFFF     FFFFFFFFFFFF
RRR    RRR       UUU          UUU  NNN      NNNNNN  000         000  FFF              FFF
RRR    RRR       UUU          UUU  NNN      NNNNNN  000         000  FFF              FFF
RRR    RRR       UUU          UUU  NNN      NNNNNN  000         000  FFF              FFF
RRR       RRR    UUU          UUU  NNN         NNN  000         000  FFF              FFF
RRR       RRR    UUU          UUU  NNN         NNN  000         000  FFF              FFF
RRR       RRR    UUU          UUU  NNN         NNN  000         000  FFF              FFF
RRR          RRR UUUUUUUUUUUUUUU  NNN         NNN     000000000     FFF              FFF
RRR          RRR UUUUUUUUUUUUUUU  NNN         NNN     000000000     FFF              FFF
RRR          RRR UUUUUUUUUUUUUUU  NNN         NNN     000000000     FFF              FFF
RRR          RRR UUUUUUUUUUUUUUU  NNN         NNN     000000000     FFF              FFF
RRR          RRR UUUUUUUUUUUUUUU  NNN         NNN     000000000     FFF              FFF
```

FILEID**NDXOUT

```
NN      NN  DDDDDDDD    XX      XX    OOOOOO    UU      UU  TTTTTTTTTT
NN      NN  DDDDDDDD    XX      XX    OOOOOO    UU      UU  TTTTTTTTTT
NN      NN  DD      DD  XX      XX  OO      OO  UU      UU      TT
NN      NN  DD      DD  XX      XX  OO      OO  UU      UU      TT
NNNN    NN  DD      DD    XX  XX    OO      OO  UU      UU      TT
NNNN    NN  DD      DD    XX  XX    OO      OO  UU      UU      TT
NN  NN  NN  DD      DD      XX      OO      OO  UU      UU      TT
NN  NN  NN  DD      DD      XX      OO      OO  UU      UU      TT
NN    NNNN  DD      DD    XX  XX    OO      OO  UU      UU      TT
NN    NNNN  DD      DD    XX  XX    OO      OO  UU      UU      TT
NN      NN  DD      DD  XX      XX  OO      OO  UU      UU      TT          ....
NN      NN  DD      DD  XX      XX  OO      OO  UU      UU      TT          ....
NN      NN  DDDDDDDD    XX      XX    OOOOOO    UUUUUUUUUU      TT          ....
NN      NN  DDDDDDDD    XX      XX    OOOOOO    UUUUUUUUUU      TT          ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

```
;     1          0001  0 %TITLE 'NDXOUT -- Sort and store index entries'
;     2          0002  0 MODULE NDXOUT (IDENT = 'V04-000'
;     3          0003  0                %BLISS32 [, ADDRESSING_MODE (EXTERNAL = LONG_RELATIVE, NONEXTERNAL = LONG_RELATIVE)]
;     4          0004  0                ) =
;     5          0005  1 BEGIN
;     6          0006  1
;     7          0007  1 !
;     8          0008  1 !*****************************************************************************
;     9          0009  1 !*                                                                           *
;    10          0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;    11          0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    12          0012  1 !*  ALL RIGHTS RESERVED.                                                      *
;    13          0013  1 !*                                                                           *
;    14          0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    15          0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;    16          0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
;    17          0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;    18          0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
;    19          0019  1 !*  TRANSFERRED.                                                              *
;    20          0020  1 !*                                                                           *
;    21          0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;    22          0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;    23          0023  1 !*  CORPORATION.                                                              *
;    24          0024  1 !*                                                                           *
;    25          0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
;    26          0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    27          0027  1 !*                                                                           *
;    28          0028  1 !*                                                                           *
;    29          0029  1 !*****************************************************************************
;    30          0030  1 !
;    31          0031  1 !++
;    32          0032  1 ! FACILITY:
;    33          0033  1 !   DSR (Digital Standard RUNOFF) /DSRPLUS DSRINDEX/INDEX Utility
;    34          0034  1 !
;    35          0035  1 ! ABSTRACT:
;    36          0036  1 !       The routines contained in this module sort and store
;    37          0037  1 !       index entries.  This module is part of INDEX and was
;    38          0038  1 !       adopted from the TCX module XOUT.
;    39          0039  1 !
;    40          0040  1 !
;    41          0041  1 ! ENVIRONMENT:  Transportable
;    42          0042  1 !
;    43          0043  1 ! AUTHOR:       JPK
;    44          0044  1 !
;    45          0045  1 ! MODIFIED BY:
;    46          0046  1 !
;    47          0047  1 !       007     JPK00018        09-Mar-1983
;    48          0048  1 !               Modified INDEX to handle new BRN format.
;    49          0049  1 !               Modified NDXOUT to handle specifyable levels on SORT= string.
;    50          0050  1 !               Modified NDXFMT to output new RUNOFF prologue.
;    51          0051  1 !               Modified NDXPAG to output new TMS prologue and RUNOFF epilogue.
;    52          0052  1 !
;    53          0053  1 !       006     JPK00015        04-Feb-1983
;    54          0054  1 !               Cleaned up module names, modified revision history to
;    55          0055  1 !               conform with established standards. Updated copyright dates.
;    56          0056  1 !
;    57          0057  1 !       005     JPK00012        24-Jan-1983
```

```
;    58          0058  1 !               Modified NDXVMSMSG.MSG to define error messages for both
;    59          0059  1 !               DSRINDEX and INDEX.
;    60          0060  1 !               Added require of NDXVMSREQ.R32 to NDXOUT, NDXFMT, NDXDAT,
;    61          0061  1 !               INDEX, NDXMSG, NDXXTN, NDXTMS, NDXVMS and NDXPAG for BLISS32.
;    62          0062  1 !               Since this file defines the error message literals,
;    63          0063  1 !               the EXTERNAL REFERENCEs for the error message literals
;    64          0064  1 !               have been removed.
;    65          0065  1 !
;    66          0066  1 !       004     JPK00010        24-Jan-1983
;    67          0067  1 !               Removed routines GETDAT and UPDDAT from NDXDAT - they
;    68          0068  1 !               performed no useful function. Removed references to these
;    69          0069  1 !               routines from NDXOUT, NDXFMT, and NDXMSG.
;    70          0070  1 !               Removed reference to XPOOL in NDXOUT - not used.
;    71          0071  1 !
;    72          0072  1 !       003     JPK00009        24-Jan-1983
;    73          0073  1 !               Modified to enhance performance. The sort buckets have each
;    74          0074  1 !               been divided into 27 sub-buckets; 1 for each letter and 1
;    75          0075  1 !               for non-alphas. Removed reference to BUCKET from INDEX.
;    76          0076  1 !               Definition of the structure was added to NDXPOL. References
;    77          0077  1 !               to BUCKET were changed in modules NDXOUT, NDXINI, NDXFMT
;    78          0078  1 !               and NDXDAT.
;    79          0079  1 !
;    80          0080  1 !       002     JPK00004        24-Sep-1982
;    81          0081  1 !               Modified NDXOUT, NDXMSG, NDXFMT, and NDXDAT for TOPS-20.
;    82          0082  1 !               Strings stored in the index pool use the first fullword
;    83          0083  1 !               for their length. References to these strings were incorrect.
;    84          0084  1 !
;    85          0085  1 !-
```

```
;    87          0086  1 %SBTTL 'Declarations'
;    88          0087  1 !
;    89          0088  1 ! TABLE OF CONTENTS:
;    90          0089  1 !
;    91          0090  1
;    92          0091  1 FORWARD ROUTINE
;    93          0092  1     XOUT        : NOVALUE,          ! Put away index item
;    94          0093  1     SORT_AS     : NOVALUE,          ! Build sort string
;    95          0094  1     FIND_POS    : NOVALUE,          ! Locate position for insertion
;    96          0095  1     FIND_BUCKET,                    ! Locate bucket for insertion
;    97          0096  1     INSERT_INX  : NOVALUE,          ! Insert index item into list
;    98          0097  1     INSERT_REF,                     ! Insert page reference into list
;    99          0098  1     ENTRY_CMP,                      ! Compare new entry with current entry
;   100          0099  1     STRG_CMP,                       ! Compare two strings
;   101          0100  1     CHRCMP      : NOVALUE;          ! Compare two characters in internal format
;   102          0101  1
;   103          0102  1 !
;   104          0103  1 ! INCLUDE FILES:
;   105          0104  1 !
;   106          0105  1
;   107          0106  1 LIBRARY 'NXPORT:XPORT';
;   108          0107  1
;   109          0108  1 SWITCHES LIST (REQUIRE);
;   110          0109  1
;   111          0110  1 REQUIRE 'REQ:NDXCLI';
```

```
;  R0111  1     !                          IDENT = 0V04-00004
;  R0112  1     !
;  R0113  1     !*****************************************************************************
;  R0114  1     !*                                                                           *
;  R0115  1     !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;  R0116  1     !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;  R0117  1     !*  ALL RIGHTS RESERVED.                                                     *
;  R0118  1     !*                                                                           *
;  R0119  1     !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
;  R0120  1     !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
;  R0121  1     !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
;  R0122  1     !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
;  R0123  1     !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
;  R0124  1     !*  TRANSFERRED.                                                             *
;  R0125  1     !*                                                                           *
;  R0126  1     !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
;  R0127  1     !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
;  R0128  1     !*  CORPORATION.                                                             *
;  R0129  1     !*                                                                           *
;  R0130  1     !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
;  R0131  1     !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;  R0132  1     !*                                                                           *
;  R0133  1     !*                                                                           *
;  R0134  1     !*****************************************************************************
;  R0135  1     !
;  R0136  1
;  R0137  1     !++
;  R0138  1     ! FACILITY:
;  R0139  1     !   DSR (Digital Standard RUNOFF) /DSRPLUS DSRINDEX/INDEX Utility
;  R0140  1     !
;  R0141  1     ! ABSTRACT:       INDEX command line definitions
;  R0142  1     !
;  R0143  1     ! ENVIRONMENT:    Transportable
;  R0144  1     !
;  R0145  1     ! AUTHOR:         JPK
;  R0146  1     !
;  R0147  1     ! CREATION DATE: January 1982
;  R0148  1     !
;  R0149  1     ! MODIFIED BY:
;  R0150  1     !
;  R0151  1     !       004     JPK00015        04-Feb-1983
;  R0152  1     !               Cleaned up module names, modified revision history to
;  R0153  1     !               conform with established standards. Updated copyright dates.
;  R0154  1     !
;  R0155  1     !       003     JPK00011        24-Jan-1983
;  R0156  1     !               Changed CMDBLK [NDX$G_LEVEL] to CMDBLK [NDX$H_LEVEL]
;  R0157  1     !               Changed CMDBLK [NDX$H_FORMAT] to CMDBLK [NDX$H_LAYOUT]
;  R0158  1     !               Changed CMDBLK [NDX$V_TMS11] and CMDBLK [NDX$V_TEX] to CMDBLK [NDX$H_FORMAT]
;  R0159  1     !               Changed comparisons of (.CHRSIZ EQLA CHRSZA) to
;  R0160  1     !               (.CMDBLK [NDX$H_FORMAT] EQL TMS11_A).
;  R0161  1     !               Definitions were changed in NDXCLI and references to the
;  R0162  1     !               effected fields were changed in NDXPAG, NDXFMT, INDEX, NDXVMS
;  R0163  1     !               and NDXCLIDMP.
;  R0164  1     !
;  R0165  1     !       002     RER00002        20-Jan-1983
;  R0166  1     !               Modified VMS command line interface module NDXVMS:
;  R0167  1     !                 - changed /FORMAT qualifier to /LAYOUT.
```

```
;  R0168  1        !                  - changed use of /RESERVE and /REQUIRE for DSRPLUS.
;  R0169  1        !                  - added code for new DSRPLUS qualifiers /FORMAT and
;  R0170  1        !                    /TELLTALE_HEADINGS.
;  R0171  1        !                Added fields to NDXCLI for new qualifiers: NDX$V_TELLTALE
;  R0172  1        !                and NDX$V_TEX.
;  R0173  1        !                Conditionalized output of NDX$V_PAGE_MERGE in NDXCLIDMP to
;  R0174  1        !                account for different DSR and DSRPLUS default values.
;  R0175  1        !
;  R0176  1        !--
;  R0177  1
```

```
;  R0178  1     !
;  R0179  1     !  NDXCMD_FIELDS
;  R0180  1     !
;  R0181  1     $FIELD ndxcmd_fields =
;  R0182  1         SET
;  R0183  1
;  R0184  1         NDX$V_OPTIONS       = [$INTEGER],          ! Command option indicators:
;  R0185  1
;  R0186  1             $OVERLAY (NDX$V_OPTIONS)
;  R0187  1
;  R0188  1             NDX$V_INPUT_CONCAT      = [$BIT],      ! Input file concatenated to previous
;  R0189  1             NDX$V_OUTPUT            = [$BIT],      ! Generate output file
;  R0190  1             NDX$V_REQUIRE           = [$BIT],      ! Require file specified
;  R0191  1             NDX$V_PAGES             = [$BIT],      ! Include page references in index
;  R0192  1             NDX$V_OVERRIDE          = [$BIT],      ! Override master index information
;  R0193  1             NDX$V_STANDARD_PAGE     = [$BIT],      ! Generate standard page numbers
;  R0194  1             NDX$V_CONTINUATION      = [$BIT],      ! Generate continuation headings
;  R0195  1             NDX$V_GUIDE             = [$BIT],      ! Generate guide headings
;  R0196  1             NDX$V_WORD_SORT         = [$BIT],      ! Sort entries word by word
;  R0197  1             NDX$V_LOG               = [$BIT],      ! Generate /LOG message
;  R0198  1             NDX$V_MASTER            = [$BIT],      ! Generate a master index
;  R0199  1             NDX$V_PAGE_MERGE        = [$BIT],      ! Merge adjacent page references
;  R0200  1             NDX$V_TELLTALE          = [$BIT],      ! Generate telltale headings
;  R0201  1
;  R0202  1             $CONTINUE
;  R0203  1
;  R0204  1         NDX$H_FORMAT            = [$SHORT_INTEGER], ! Output format: DSR, TMS, TEX
;  R0205  1         NDX$H_LAYOUT            = [$SHORT_INTEGER], ! Output layout type
;  R0206  1         NDX$H_NONALPHA          = [$SHORT_INTEGER], ! Treatment of leading nonalphas during sort
;  R0207  1         NDX$H_LEVEL             = [$SHORT_INTEGER], ! Deepest level to include in index
;  R0208  1         NDX$G_COLUMN_WID        = [$INTEGER],       ! Column width
;  R0209  1         NDX$G_GUTTER_WID        = [$INTEGER],       ! Gutter width
;  R0210  1         NDX$G_LINES_PAGE        = [$INTEGER],       ! Lines per page
;  R0211  1         NDX$G_RESERVE_LINES     = [$INTEGER],       ! Number of lines to reserve when requiring a file
;  R0212  1         NDX$G_SEPARATE_WIDTH    = [$INTEGER],       ! Width of reference portion of entry
;  R0213  1         NDX$T_MASTER_BOOK       = [$DESCRIPTOR(DYNAMIC)], ! Book name descriptor for Master indexing
;  R0214  1         NDX$T_INPUT_FILE        = [$DESCRIPTOR(DYNAMIC)], ! Input file name descriptor
;  R0215  1         NDX$T_OUTPUT_FILE       = [$DESCRIPTOR(DYNAMIC)], ! Output file name descriptor
;  R0216  1         NDX$T_REQUIRE_FILE      = [$DESCRIPTOR(DYNAMIC)], ! Require file name descriptor
;  R0217  1         NDX$T_RELATED_FILE      = [$DESCRIPTOR(DYNAMIC)], ! Related file name descriptor is saved here
;  R0218  1                                                           ! by NDXINP for later use by MAKNDX
;  R0219  1         NDX$T_COMMAND_LINE      = [$DESCRIPTOR(DYNAMIC)]  ! Copy of entire command line
;  R0220  1
;  R0221  1         TES;
;  R0222  1     !
;  R0223  1     !  End of NDXCMD_FIELDS
;  R0224  1     !
;  R0225  1
;  R0226  1     LITERAL
;  R0227  1         NDXCMD$K_LENGTH = $FIELD_SET_SIZE;
;  R0228  1
;  R0229  1     MACRO
;  R0230  1         $NDXCMD = BLOCK [NDXCMD$K_LENGTH] FIELD (NDXCMD_FIELDS) %;
;  R0231  1
;  R0232  1     $LITERAL                                        ! Output formats (NDX$H_FORMAT)
;  R0233  1         DSR                 = $DISTINCT,           ! Runoff
;  R0234  1         TMS11_A             = $DISTINCT,           ! TMS=A
```

```
;  R0235  1          TMS11_E                 = $DISTINCT,    ! TMS=E
;  R0236  1          TEX                     = $DISTINCT;    ! TEX
;  R0237  1
;  R0238  1      $LITERAL                                    ! Output layouts (NDX$H_LAYOUT)
;  R0239  1          TWO_COLUMN              = $DISTINCT,    ! Normal two column format
;  R0240  1          ONE_COLUMN              = $DISTINCT,    ! Normal one column format
;  R0241  1          SEPARATE                = $DISTINCT,    ! Separate reference format
;  R0242  1          GALLEY                  = $DISTINCT;    ! TMS11 Galley format
;  R0243  1
;  R0244  1      $LITERAL                                    ! Treatment of leading nonalphas during sort (NDX$H_NONALPHA)
;  R0245  1          BEFORE                  = $DISTINCT,    ! Leading nonalphas sort before alphas
;  R0246  1          AFTER                   = $DISTINCT,    ! Leading nonalphas sort after alphas
;  R0247  1          IGNORE                  = $DISTINCT;    ! Leading nonalphas are ignored
;  R0248  1
;  R0249  1      !
;  R0250  1      !--     End of NDXCLI.REQ
```

```
;  112          0251  1
;  113          0252  1 REQUIRE 'REQ:NDXXPL';
```

```
;  R0253  1     !
;  R0254  1     ! Version:      'V04-000'
;  R0255  1     !
;  R0256  1     !****************************************************************************
;  R0257  1     !*                                                                          *
;  R0258  1     !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;  R0259  1     !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
;  R0260  1     !*  ALL RIGHTS RESERVED.                                                    *
;  R0261  1     !*                                                                          *
;  R0262  1     !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;  R0263  1     !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
;  R0264  1     !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;  R0265  1     !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;  R0266  1     !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;  R0267  1     !*  TRANSFERRED.                                                            *
;  R0268  1     !*                                                                          *
;  R0269  1     !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;  R0270  1     !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;  R0271  1     !*  CORPORATION.                                                            *
;  R0272  1     !*                                                                          *
;  R0273  1     !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;  R0274  1     !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
;  R0275  1     !*                                                                          *
;  R0276  1     !*                                                                          *
;  R0277  1     !****************************************************************************
;  R0278  1     !
;  R0279  1     !
;  R0280  1     !++
;  R0281  1     ! FACILITY:      DSR (Digital Standard RUNOFF) / DSRPLUS
;  R0282  1     !
;  R0283  1     ! ABSTRACT:
;  R0284  1     !       This file contains definitions of data structures used to support
;  R0285  1     !       the extended indexing features of the DSRPLUS INDEX program.
;  R0286  1     !
;  R0287  1     ! ENVIRONMENT:   Transportable BLISS
;  R0288  1     !
;  R0289  1     ! AUTHOR:        J.P. Kellerman
;  R0290  1     !
;  R0291  1     ! CREATION DATE: January 1982
;  R0292  1     !
;  R0293  1     ! MODIFIED BY:
;  R0294  1     !
;  R0295  1     !       002     KAD00002        Keith Dawson    07-Mar-1983
;  R0296  1     !               Global edit of all modules. Updated module names, idents,
;  R0297  1     !               copyright dates. Changed require files to BLISS library.
;  R0298  1     !
;  R0299  1     !--
;  R0300  1     !
;  R0301  1     ! Extended INDEX attributes block.
;  R0302  1     !
;  R0303  1     $FIELD XPL_FIELDS =
;  R0304  1         SET
;  R0305  1     !
;  R0306  1         XPL$V_OPTIONS       = [$INTEGER],           ! Attributes options
;  R0307  1     !
;  R0308  1             $OVERLAY (XPL$V_OPTIONS)
;  R0309  1
```

```
;   R0310  1              XPL$V_VALID             = [$BIT],       ! Attributes block contains valid information.
;   R0311  1              XPL$V_BOLD              = [$BIT],       ! Bold page reference.
;   R0312  1              XPL$V_UNDERLINE         = [$BIT],       ! Underlined page reference.
;   R0313  1              XPL$V_BEGIN             = [$BIT],       ! Begin page range.
;   R0314  1              XPL$V_END               = [$BIT],       ! End page range.
;   R0315  1              XPL$V_MASTER            = [$BIT],       ! Master index entry.
;   R0316  1              XPL$V_PERMUTE           = [$BIT],       ! Permute index entry.
;   R0317  1              XPL$V_NOPERMUTE         = [$BIT],       ! Set if permute explicitly forbidden.
;   R0318  1              XPL$V_SORT              = [$BIT],       ! Set if SORT string present.
;   R0319  1              XPL$V_APPEND            = [$BIT],       ! Set if append string present.
;   R0320  1
;   R0321  1              $CONTINUE
;   R0322  1
;   R0323  1          XPL$T_SORT          = [$DESCRIPTOR(DYNAMIC)], ! SORT string.
;   R0324  1          XPL$T_APPEND        = [$DESCRIPTOR(DYNAMIC)]  ! APPEND string.
;   R0325  1
;   R0326  1          TES;
;   R0327  1
;   R0328  1      LITERAL
;   R0329  1          XPL$K_LENGTH = $FIELD_SET_SIZE;
;   R0330  1
;   R0331  1      MACRO
;   R0332  1          $XPL_BLOCK = BLOCK [XPL$K_LENGTH] FIELD (XPL_FIELDS) %;
;   R0333  1
;   R0334  1      !
;   R0335  1      ! Macros for INDEX_ATTRIBUTES flags
;   R0336  1      !
;   R0337  1      MACRO
;   R0338  1          XPLUS$V_VALID       = 0, 0, 1, 0 %,  ! Set if attributes data is valid.
;   R0339  1          XPLUS$V_BOLD        = 0, 1, 1, 0 %,  ! Set if page reference is bolded.
;   R0340  1          XPLUS$V_UNDERLINE   = 0, 2, 1, 0 %,  ! Set if page reference is underlined.
;   R0341  1          XPLUS$V_BEGIN       = 0, 3, 1, 0 %,  ! Set if entry begins a page range.
;   R0342  1          XPLUS$V_END         = 0, 4, 1, 0 %,  ! Set if entry ends a page range.
;   R0343  1          XPLUS$V_MASTER      = 0, 5, 1, 0 %,  ! Set if master index entry only.
;   R0344  1          XPLUS$V_PERMUTE     = 0, 6, 1, 0 %,  ! Set if entry is to be permuted.
;   R0345  1          XPLUS$V_NOPERMUTE   = 0, 7, 1, 0 %,  ! Set if permute is explicitly forbidden.
;   R0346  1          XPLUS$V_SORT        = 0, 8, 1, 0 %,  ! Set if entry contains a SORT string.
;   R0347  1          XPLUS$V_APPEND      = 0, 9, 1, 0 %;  ! Set if entry contains an APPEND string.
;   R0348  1
;   R0349  1      !                           End of NDXXPL.REQ
```

```
;  114          0350  1
;  115          0351  1 REQUIRE 'REQ:NDXPOL';
```

```
;  R0352  1        !
;  R0353  1        ! Version:      'V04-000'
;  R0354  1        !
;  R0355  1        !****************************************************************************
;  R0356  1        !*                                                                          *
;  R0357  1        !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;  R0358  1        !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
;  R0359  1        !*  ALL RIGHTS RESERVED.                                                    *
;  R0360  1        !*                                                                          *
;  R0361  1        !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;  R0362  1        !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;  R0363  1        !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;  R0364  1        !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;  R0365  1        !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;  R0366  1        !*  TRANSFERRED.                                                            *
;  R0367  1        !*                                                                          *
;  R0368  1        !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;  R0369  1        !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;  R0370  1        !*  CORPORATION.                                                            *
;  R0371  1        !*                                                                          *
;  R0372  1        !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;  R0373  1        !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
;  R0374  1        !*                                                                          *
;  R0375  1        !*                                                                          *
;  R0376  1        !****************************************************************************
;  R0377  1        !
;  R0378  1
;  R0379  1        !++
;  R0380  1        ! FACILITY:
;  R0381  1        !   DSR (Digital Standard RUNOFF) /DSRPLUS DSRINDEX/INDEX Utility
;  R0382  1        !
;  R0383  1        ! ABSTRACT:
;  R0384  1        !   This file contains literals and macros defining the data structures
;  R0385  1        !   found in the internal index pool
;  R0386  1        !
;  R0387  1        ! ENVIRONMENT:   Transportable
;  R0388  1        !
;  R0389  1        ! AUTHOR:        JPK
;  R0390  1        !
;  R0391  1        ! CREATION DATE: January 1982
;  R0392  1        !
;  R0393  1        ! MODIFIED BY:
;  R0394  1        !
;  R0395  1        !       003     JPK00015        04-Feb-1983
;  R0396  1        !               Cleaned up module names, modified revision history to
;  R0397  1        !               conform with established standards. Updated copyright dates.
;  R0398  1        !
;  R0399  1        !       002     JPK00009        24-Jan-1983
;  R0400  1        !               Modified to enhance performance. The sort buckets have each
;  R0401  1        !               been divided into 27 sub-buckets; 1 for each letter and 1
;  R0402  1        !               for non-alphas. Removed reference to BUCKET from INDEX.
;  R0403  1        !               Definition of the structure was added to NDXPOL. References
;  R0404  1        !               to BUCKET were changed in modules NDXOUT, NDXINI, NDXFMT
;  R0405  1        !               and NDXDAT.
;  R0406  1        !
;  R0407  1        !--
;  R0408  1
```

```
;  R0409  1     ! Index entry
;  R0410  1
;  R0411  1     $FIELD XE_FIELDS =
;  R0412  1         SET
;  R0413  1
;  R0414  1         XE$A_PREV           = [$ADDRESS],       ! Link to previous item
;  R0415  1         XE$A_NEXT           = [$ADDRESS],       ! Link to next item
;  R0416  1         XE$A_SUBX           = [$ADDRESS],       ! Sub index pointer
;  R0417  1         XE$A_REF            = [$ADDRESS],       ! Reference pointer
;  R0418  1         XE$A_TEXT           = [$ADDRESS],       ! Pointer to text of index item
;  R0419  1         XE$A_SORT_AS        = [$ADDRESS],       ! Pointer to SORT_AS string
;  R0420  1         XE$H_SUBC           = [$SHORT_INTEGER], ! Sub index level
;  R0421  1
;  R0422  1         XE$V_FLAGS          = [$SHORT_INTEGER], ! Entry flags
;  R0423  1
;  R0424  1             $OVERLAY (XE$V_FLAGS)
;  R0425  1
;  R0426  1             XE$V_BARS               = [$BIT],   ! Change bar flag
;  R0427  1
;  R0428  1             $CONTINUE
;  R0429  1
;  R0430  1         XE$A_BOOK_LIST      = [$ADDRESS]        ! Master index book name list
;  R0431  1
;  R0432  1         $ALIGN (FULLWORD)
;  R0433  1
;  R0434  1         TES;
;  R0435  1
;  R0436  1     LITERAL
;  R0437  1         XE$K_LENGTH = $FIELD_SET_SIZE;
;  R0438  1
;  R0439  1     MACRO
;  R0440  1         $XE_BLOCK = BLOCK [XE$K_LENGTH] FIELD (XE_FIELDS) %;
;  R0441  1
;  R0442  1     ! End of Index entry
;  R0443  1
;  R0444  1
;  R0445  1     ! Reference entry
;  R0446  1
;  R0447  1     $FIELD XX_FIELDS =
;  R0448  1         SET
;  R0449  1
;  R0450  1         XX$A_LINK           = [$ADDRESS],       ! Link to additional entries
;  R0451  1         XX$A_APPEND         = [$ADDRESS],       ! APPEND text pointer
;  R0452  1         XX$H_PAGE           = [$SHORT_INTEGER], ! Transaction number
;  R0453  1
;  R0454  1         XX$V_FLAGS          = [$SHORT_INTEGER], ! Display attributes
;  R0455  1
;  R0456  1             $OVERLAY (XX$V_FLAGS)
;  R0457  1
;  R0458  1             XX$V_BOLD       = [$BIT],           ! Bold page reference
;  R0459  1             XX$V_UNDERLINE  = [$BIT],           ! Underline page reference
;  R0460  1             XX$V_BEGIN      = [$BIT],           ! Begin page range
;  R0461  1             XX$V_END        = [$BIT],           ! End page range
;  R0462  1
;  R0463  1             $CONTINUE
;  R0464  1
;  R0465  1         XX$A_BOOK           = [$ADDRESS]        ! Master index book name
```

```
;  R0466  1
;  R0467  1        $ALIGN (FULLWORD)
;  R0468  1
;  R0469  1        TES;
;  R0470  1
;  R0471  1    LITERAL
;  R0472  1        XX$K_LENGTH = $FIELD_SET_SIZE;
;  R0473  1
;  R0474  1    MACRO
;  R0475  1        $XX_BLOCK = BLOCK [XX$K_LENGTH] FIELD (XX_FIELDS) %;
;  R0476  1
;  R0477  1    ! End of Reference entry
;  R0478  1
;  R0479  1
;  R0480  1    ! Master index book reference entry
;  R0481  1
;  R0482  1    $FIELD XM_FIELDS =
;  R0483  1        SET
;  R0484  1
;  R0485  1        XM$A_LINK           = [$ADDRESS],       ! Link to additional entries
;  R0486  1        XM$A_BOOK           = [$ADDRESS]        ! Pointer to book name
;  R0487  1
;  R0488  1        TES;
;  R0489  1
;  R0490  1    LITERAL
;  R0491  1        XM$K_LENGTH = $FIELD_SET_SIZE;
;  R0492  1
;  R0493  1    MACRO
;  R0494  1        $XM_BLOCK = BLOCK [XM$K_LENGTH] FIELD (XM_FIELDS) %;
;  R0495  1
;  R0496  1    ! End of Master index book reference entry
;  R0497  1
;  R0498  1
;  R0499  1    ! Current Entry
;  R0500  1
;  R0501  1    $FIELD C_FIELDS =
;  R0502  1        SET
;  R0503  1
;  R0504  1        C$A_CURR            = [$ADDRESS],       ! Pointer to current cell
;  R0505  1        C$A_PREV            = [$ADDRESS],       ! Pointer to previous cell
;  R0506  1        C$A_HEAD            = [$ADDRESS],       ! Pointer to head of chain
;  R0507  1
;  R0508  1        $ALIGN (FULLWORD)
;  R0509  1
;  R0510  1        C$V_FLAGS           = [$INTEGER],       ! Current cell flags
;  R0511  1
;  R0512  1            $OVERLAY (C$V_FLAGS)
;  R0513  1
;  R0514  1            C$V_IDNS        = [$BIT]            ! Identical string flag
;  R0515  1
;  R0516  1            $CONTINUE
;  R0517  1
;  R0518  1        TES;
;  R0519  1
;  R0520  1    LITERAL
;  R0521  1        C$K_LENGTH = $FIELD_SET_SIZE;
;  R0522  1
```

```
;  R0523  1        MACRO
;  R0524  1            $C_BLOCK = BLOCK [C$K_LENGTH] FIELD (C_FIELDS) %;
;  R0525  1
;  R0526  1        ! End of current entry
;  R0527  1
;  R0528  1
;  R0529  1        !
;  R0530  1        ! Dummy datasets
;  R0531  1        !
;  R0532  1        LITERAL
;  R0533  1            DS_X_ENTRY  = XE$K_LENGTH,
;  R0534  1            DS_XX_ENTRY = XX$K_LENGTH,
;  R0535  1            DS_XM_ENTRY = XM$K_LENGTH,
;  R0536  1            DS_X_STRING = 0;
;  R0537  1
;  R0538  1        !
;  R0539  1        ! Structure definition for bucket array.
;  R0540  1        !
;  R0541  1        !       Buckets are arranged so that each row represents the first letter of
;  R0542  1        !       the string and each column represents the second letter of the string.
;  R0543  1        !
;  R0544  1        !       This approach is used only for master indexes as no performance
;  R0545  1        !       improvement is realised until about 10 input files have been processed.
;  R0546  1        !
;  R0547  1        !       Indexes which are not master indexes use only the first element of
;  R0548  1        !       each row, i.e., [0, 0] ... [26, 0].
;  R0549  1        !
;  R0550  1        !       The only exception is for nonalphabetic characters which use only
;  R0551  1        !       element [0, 0]. Elements [0, 1] ... [0, 26] are not used since mapping
;  R0552  1        !       all nonalphabetics into one row loses the sort order of the first
;  R0553  1        !       character in the string. For nonalphabetics to work correctly in a two
;  R0554  1        !       dimensional bucket scheme, the array would have to be at least 127 x 127
;  R0555  1        !
;  R0556  1        !             0   1   .   .   .  26
;  R0557  1        !          0  **  not used  .   .   .
;  R0558  1        !          1  A?  AA            AZ
;  R0559  1        !          .                    .
;  R0560  1        !          .                    .
;  R0561  1        !          .                    .
;  R0562  1        !         26  Z?  ZA   .   .   .  ZZ
;  R0563  1        !
;  R0564  1        STRUCTURE
;  R0565  1            $BUCKET_ARRAY [ROW_IDX, COL_IDX; M, N] =
;  R0566  1                [M * N * %UPVAL] ($BUCKET_ARRAY + (ROW_IDX * N + COL_IDX) * %UPVAL);
;  R0567  1
;  R0568  1        !--     End of NDXPOL.REQ
```

```
;  116        0569  1
;  117        0570  1 REQUIRE 'REQ:LETTER';
```

```
;  R0571  1        !
;  R0572  1        ! Version:      'V04-000'
;  R0573  1        !
;  R0574  1        !****************************************************************************
;  R0575  1        !*                                                                          *
;  R0576  1        !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                *
;  R0577  1        !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
;  R0578  1        !*  ALL RIGHTS RESERVED.                                                    *
;  R0579  1        !*                                                                          *
;  R0580  1        !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;  R0581  1        !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;  R0582  1        !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;  R0583  1        !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;  R0584  1        !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;  R0585  1        !*  TRANSFERRED.                                                            *
;  R0586  1        !*                                                                          *
;  R0587  1        !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;  R0588  1        !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;  R0589  1        !*  CORPORATION.                                                            *
;  R0590  1        !*                                                                          *
;  R0591  1        !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;  R0592  1        !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
;  R0593  1        !*                                                                          *
;  R0594  1        !*                                                                          *
;  R0595  1        !****************************************************************************
;  R0596  1        !
;  R0597  1
;  R0598  1        !++
;  R0599  1        ! FACILITY:      DSR (Digital Standard RUNOFF) / DSRPLUS
;  R0600  1        !
;  R0601  1        ! ABSTRACT:
;  R0602  1        !       Macros to test if a character is an appropriately flavored letter,
;  R0603  1        !       and macros to convert between upper and lower case.
;  R0604  1        !
;  R0605  1        ! ENVIRONMENT:   Transportable BLISS
;  R0606  1        !
;  R0607  1        ! AUTHOR:        Rich Friday
;  R0608  1        !
;  R0609  1        ! CREATION DATE: 1978
;  R0610  1        !
;  R0611  1        ! MODIFIED BY:
;  R0612  1        !
;  R0613  1        !       002     KAD00002        Keith Dawson    07-Mar-1983
;  R0614  1        !               Global edit of all modules. Updated module names, idents,
;  R0615  1        !               copyright dates. Changed require files to BLISS library.
;  R0616  1        !
;  R0617  1        !--
;  R0618  1
;  R0619  1        MACRO
; MR0620  1            upper_letter (khar) =      ! See if upper case letter
; MR0621  1                (Khar GEQ %C'A' AND khar LEQ %C'Z')
;  R0622  1            %,
;  R0623  1
; MR0624  1            lower_letter (khar) =      ! See if lower case letter
; MR0625  1                (Khar GEQ %C'a' AND khar LEQ %C'z')
;  R0626  1            %,
;  R0627  1
```

```
;  MR0628  1          letter (khar)                   =       ! See if any type of letter
;  MR0629  1              (upper_letter (khar) OR lower_letter (khar))
;    R0630  1          %;
;    R0631  1
;    R0632  1      MACRO
;  MR0633  1          upper_case (khar)   =           ! Convert to upper case
;  MR0634  1              (khar + %C'A' - %C'a')
;    R0635  1          %,
;    R0636  1
;  MR0637  1          lower_case (khar)   =           ! Convert to lower case
;  MR0638  1              (khar + %C'a' - %C'A')
;    R0639  1          %;
;    R0640  1
;    R0641  1      !                           End of LETTER.REQ
```

```
;  118          0642  1
;  119        L 0643  1 %IF %BLISS (BLISS32)
;  120          0644  1 %THEN
;  121          0645  1
;  122          0646  1 REQUIRE 'REQ:NDXVMSREQ';
```

```
;  R0647  1        !
;  R0648  1        ! Version:      'V04-000'
;  R0649  1        !
;  R0650  1        !****************************************************************************
;  R0651  1        !*                                                                          *
;  R0652  1        !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;  R0653  1        !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
;  R0654  1        !*  ALL RIGHTS RESERVED.                                                    *
;  R0655  1        !*                                                                          *
;  R0656  1        !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;  R0657  1        !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;  R0658  1        !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;  R0659  1        !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;  R0660  1        !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;  R0661  1        !*  TRANSFERRED.                                                            *
;  R0662  1        !*                                                                          *
;  R0663  1        !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;  R0664  1        !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;  R0665  1        !*  CORPORATION.                                                            *
;  R0666  1        !*                                                                          *
;  R0667  1        !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;  R0668  1        !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
;  R0669  1        !*                                                                          *
;  R0670  1        !*                                                                          *
;  R0671  1        !****************************************************************************
;  R0672  1        !
;  R0673  1
;  R0674  1        !++
;  R0675  1        ! FACILITY:
;  R0676  1        !   DSR (Digital Standard RUNOFF) /DSRPLUS DSRINDEX/INDEX Utility
;  R0677  1        !
;  R0678  1        ! ABSTRACT:
;  R0679  1        !       This file contains external references to the error message numbers
;  R0680  1        !       for DSRINDEX/INDEX.
;  R0681  1        !
;  R0682  1        !       New messages must be defined in NDXVMSMSG.MSG and referenced here:
;  R0683  1        !       both in the MACRO section (for DSRINDEX) and the EXTERNAL LITERAL
;  R0684  1        !       section (for INDEX)
;  R0685  1        !
;  R0686  1        ! ENVIRONMENT:   VAX/VMS User Mode
;  R0687  1        !
;  R0688  1        ! AUTHOR:        JPK
;  R0689  1        !
;  R0690  1        ! CREATION DATE: 01-Feb-1983
;  R069:  1        !
;  R0692  1        ! MODIFIED BY:
;  R0693  1        !
;  R0694  1        !       004     JPK00022        30-Mar-1983
;  R0695  1        !               Modified NDXVMS, NDXFMT, NDXPAG, NDXVMSMSG and NDXVMSREQ
;  R0696  1        !               to generate TEX output. Added module NDXTEX.
;  R0697  1        !
;  R0698  1        !       003     JPK00021        28-Mar-1983
;  R0699  1        !               Modified NDXT20 to include E2.0 functionality.
;  R0700  1        !               Modified NDXCLIDMP, NDXFMT, NDXPAG, NDXVRS to require RNODEF
;  R0701  1        !               for BLISS36 and to remove any conditional require based on
;  R0702  1        !               DSRPLUS_DEF.
;  R0703  1        !
```

```
;  R0704  1     !        002     JPK00010        04-Feb-1983
;  R0705  1     !                Cleaned up module names, modified revision history to
;  R0706  1     !                conform with established standards. Updated copyright dates.
;  R0707  1     !
;  R0708  1     !--
;  R0709  1
;  R0710  1     REQUIRE 'REQ:RNODEF';
```

```
;  R0711  1        !
;  R0712  1        ! Version:      'V04-000'
;  R0713  1        !
;  R0714  1        !****************************************************************************
;  R0715  1        !*                                                                          *
;  R0716  1        !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                *
;  R0717  1        !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
;  R0718  1        !*  ALL RIGHTS RESERVED.                                                    *
;  R0719  1        !*                                                                          *
;  R0720  1        !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;  R0721  1        !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;  R0722  1        !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;  R0723  1        !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;  R0724  1        !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;  R0725  1        !*  TRANSFERRED.                                                            *
;  R0726  1        !*                                                                          *
;  R0727  1        !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;  R0728  1        !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;  R0729  1        !*  CORPORATION.                                                            *
;  R0730  1        !*                                                                          *
;  R0731  1        !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;  R0732  1        !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
;  R0733  1        !*                                                                          *
;  R0734  1        !*                                                                          *
;  R0735  1        !****************************************************************************
;  R0736  1        !
;  R0737  1        !
;  R0738  1        !++
;  R0739  1        ! FACILITY:      DSR (Digital Standard RUNOFF) / DSRPLUS
;  R0740  1        !
;  R0741  1        ! ABSTRACT:
;  R0742  1        !       Converts BLISS/VARIANT values into useful names.
;  R0743  1        !
;  R0744  1        ! ENVIRONMENT:   Transportable BLISS
;  R0745  1        !
;  R0746  1        ! AUTHOR:        Rich Friday
;  R0747  1        !
;  R0748  1        ! CREATION DATE: 1978
;  R0749  1        !
;  R0750  1        ! MODIFIED BY:
;  R0751  1        !
;  R0752  1        !       016     KAD00016        Ray Marshall    19-Mar-1984
;  R0753  1        !               Added GERMAN, FRENCH, & ITALIAN.
;  R0754  1        !
;  R0755  1        !       015     KAD00015        Keith Dawson    18-Apr-1983
;  R0756  1        !               Made the LN01 conditional the default for vanilla DSR --
;  R0757  1        !                 its value is 0 (no variant supplied).
;  R0758  1        !
;  R0759  1        !       014     KAD00014        Keith Dawson    22-Mar-1983
;  R0760  1        !               Asserted the LN01 conditional when DSRPLUS is asserted.
;  R0761  1        !
;  R0762  1        !       013     KAD00013        Keith Dawson    20-Mar-1983
;  R0763  1        !               Removed all references to .BIX and .BTC files.
;  R0764  1        !
;  R0765  1        !       012     KAD00012        Keith Dawson    07-Mar-1983
;  R0766  1        !               Global edit of all modules. Updated module names, idents,
;  R0767  1        !               copyright dates. Changed require files to BLISS library.
```

```
;  R0768  1        !
;  R0769  1        !--
;  R0770  1
;  R0771  1        !++
;  R0772  1        !       D E F I N I T I O N   O F  /VARIANT   B I T S
;  R0773  1        !
;  R0774  1        !       The bit assignments are as follows:
;  R0775  1        !
;  R0776  1        !       Bit  Weight   Meaning
;  R0777  1        !-------------------------------------------------------------------------
;  R0778  1        !       --      0     If no /VARIANT is supplied (as for vanilla DSR),
;  R0779  1        !                     compile with LN01 support. LN01 support is also
;  R0780  1        !                     implied by the DSRPLUS variant.
;  R0781  1        !
;  R0782  1        !        0      1     CLEAR =   Unassigned
;  R0783  1        !                     SET   =   Unassigned
;  R0784  1        !
;  R0785  1        !        1      2     CLEAR =   Normal compile
;  R0786  1        !                     SET   =   Compile for DSRPLUS
;  R0787  1        !
;  R0788  1        !       4-6    16     CLEAR =   English (American) version
;  R0789  1        !                     SET   =   16 = German (Austrian)
;  R0790  1        !                               32 = French
;  R0791  1        !                               48 = Italian
;  R0792  1        !--
;  R0793  1
;  R0794  1        !-------------------------------------------------------------------------
;  R0795  1        !   This variable (LN01) controls whether or not to compile an LN01-flavored
;  R0796  1        !   DSR. It is asserted by default, and also whenever DSRPLUS is asserted.
;  R0797  1        !
;  R0798  1        !   Modules utilizing LN01 are:
;  R0799  1        !
;  R0800  1        !       DOOPTS  NOUT
;  R0801  1
;  R0802  1        COMPILETIME
;  R0803  1            ln01 =
;  R0804  2                ( (%VARIANT EQL 0) OR  %VARIANT/2 )
;  R0805  1            ;
;  R0806  1
;  R0807  1        !-------------------------------------------------------------------------
;  R0808  1        !   This variable (DSRPLUS) controls compilation for the DSRPLUS program.
;  R0809  1        !
;  R0810  1        !   All modules utilize DSRPLUS.
;  R0811  1
;  R0812  1        COMPILETIME
;  R0813  1            dsrplus =
;  R0814  2                ( %VARIANT/2 )
;  R0815  1            ;
;  R0816  1
;  R0817  1        !-------------------------------------------------------------------------
;  R0818  1        !   This variable (FLIP) controls compilation of FLIP features of DSRPLUS.
;  R0819  1        !   It assures that FLIP features are compiled only on VMS systems.
;  R0820  1        !
;  R0821  1        !   Modules utilizing FLIP are many and various.
;  R0822  1        !
;  R0823  1        COMPILETIME
;  R0824  1            flip =
```

```
;  R0825  2              ( %VARIANT/2 AND %BLISS(BLISS32) )
;  R0826  1          ;
;  R0827  1
;  R0828  1      !-----------------------------------------------------------------------
;  R0829  1      !       4-6    16     CLEAR =   English (American) version
;  R0830  1      !                     SET   =   16 = German (Austrian)
;  R0831  1      !                               32 = French
;  R0832  1      !                               48 = Italian
;  R0833  1      COMPILETIME
;  R0834  1          German  = ( %VARIANT/16 AND NOT %VARIANT/32 AND NOT %VARIANT/64 ) ;
;  R0835  1      COMPILETIME
;  R0836  1          French  = ( NOT %VARIANT/16 AND %VARIANT/32 AND NOT %VARIANT/64 ) ;
;  R0837  1      COMPILETIME
;  R0838  1          Italian = ( %VARIANT/16 AND %VARIANT/32 AND NOT %VARIANT/64 ) ;
;  R0839  1      !-----------------------------------------------------------------------
;  R0840  1      !                       End of RNODEF.REQ
```

```
;  R0841  1
; LR0842  1      %IF NOT DSRPLUS
;  R0843  1      %THEN
;  R0844  1
;  R0845  1      MACRO
;  R0846  1          INDEX$_BADLOGIC  = DSRINDEX$_BADLOGIC  %,
;  R0847  1          INDEX$_BADVALUE  = DSRINDEX$_BADVALUE  %,
;  R0848  1          INDEX$_INSVIRMEM = DSRINDEX$_INSVIRMEM %,
;  R0849  1          INDEX$_LINELENG  = DSRINDEX$_LINELENG  %,
;  R0850  1          INDEX$_NOREF     = DSRINDEX$_NOREF     %,
;  R0851  1          INDEX$_OPENIN    = DSRINDEX$_OPENIN    %,
;  R0852  1          INDEX$_OPENOUT   = DSRINDEX$_OPENOUT   %,
;  R0853  1          INDEX$_TOOMANY   = DSRINDEX$_TOOMANY   %,
;  R0854  1          INDEX$_VALERR    = DSRINDEX$_VALERR    %,
;  R0855  1          INDEX$_CANTBAL   = DSRINDEX$_CANTBAL   %,
;  R0856  1          INDEX$_CLOSEQUOT = DSRINDEX$_CLOSEQUOT %,
;  R0857  1          INDEX$_CONFQUAL  = DSRINDEX$_CONFQUAL  %,
;  R0858  1          INDEX$_CTRLCHAR  = DSRINDEX$_CTRLCHAR  %,
;  R0859  1          INDEX$_DOESNTFIT = DSRINDEX$_DOESNTFIT %,
;  R0860  1          INDEX$_DUPBEGIN  = DSRINDEX$_DUPBEGIN  %,
;  R0861  1          INDEX$_EMPTYIN   = DSRINDEX$_EMPTYIN   %,
;  R0862  1          INDEX$_IGNORED   = DSRINDEX$_IGNORED   %,
;  R0863  1          INDEX$_INVINPUT  = DSRINDEX$_INVINPUT  %,
;  R0864  1          INDEX$_INVRECORD = DSRINDEX$_INVRECORD %,
;  R0865  1          INDEX$_LASTCONT  = DSRINDEX$_LASTCONT  %,
;  R0866  1          INDEX$_NOBEGIN   = DSRINDEX$_NOBEGIN   %,
;  R0867  1          INDEX$_NOEND     = DSRINDEX$_NOEND     %,
;  R0868  1          INDEX$_NOINDEX   = DSRINDEX$_NOINDEX   %,
;  R0869  1          INDEX$_NOLIST    = DSRINDEX$_NOLIST    %,
;  R0870  1          INDEX$_OVERSTRK  = DSRINDEX$_OVERSTRK  %,
;  R0871  1          INDEX$_SKIPPED   = DSRINDEX$_SKIPPED   %,
;  R0872  1          INDEX$_SYNTAX    = DSRINDEX$_SYNTAX    %,
;  R0873  1          INDEX$_TEXFILE   = DSRINDEX$_TEXFILE   %,
;  R0874  1          INDEX$_TOODEEP   = DSRINDEX$_TOODEEP   %,
;  R0875  1          INDEX$_TOOFEW    = DSRINDEX$_TOOFEW    %,
;  R0876  1          INDEX$_TRUNCATED = DSRINDEX$_TRUNCATED %,
;  R0877  1          INDEX$_COMPLETE  = DSRINDEX$_COMPLETE  %,
;  R0878  1          INDEX$_CREATED   = DSRINDEX$_CREATED   %,
;  R0879  1          INDEX$_IDENT     = DSRINDEX$_IDENT     %,
;  R0880  1          INDEX$_PROCFILE  = DSRINDEX$_PROCFILE  %,
;  R0881  1          INDEX$_TEXT      = DSRINDEX$_TEXT      %,
;  R0882  1          INDEX$_TEXTD     = DSRINDEX$_TEXTD     %,
;  R0883  1          INDEX$_TMS11     = DSRINDEX$_TMS11     %;
;  R0884  1
;  R0885  1      %FI
;  R0886  1
;  R0887  1      EXTERNAL LITERAL
;  R0888  1          INDEX$_BADLOGIC,    ! <internal logic error detected>
;  R0889  1          INDEX$_BADVALUE,    ! <'!AS' is an invalid keyword value>
;  R0890  1          INDEX$_INSVIRMEM,   ! <insufficient virtual memory>
;  R0891  1          INDEX$_LINELENG,    ! <maximum line length is 120>
;  R0892  1          INDEX$_NOREF,       ! <page reference not found>
;  R0893  1          INDEX$_OPENIN,      ! <error opening '!AS' for input>
;  R0894  1          INDEX$_OPENOUT,     ! <error opening '!AS' for output>
;  R0895  1          INDEX$_TOOMANY,     ! <too many values supplied>
;  R0896  1          INDEX$_VALERR,      ! <specified value is out of legal range>
;  R0897  1          INDEX$_CANTBAL;     ! <can't balance last page>
```

```
;  R0898  1          INDEX$_CLOSEQUOT,   ! <missing close quote>
;  R0899  1          INDEX$_CONFQUAL,    ! <conflicting qualifiers>
;  R0900  1          INDEX$_CTRLCHAR,    ! <the following line contains control characters - ignored>
;  R0901  1          INDEX$_DOESNTFIT,   ! <'!AD' will not fit at the current indentation level>
;  R0902  1          INDEX$_DUPBEGIN,    ! <duplicate .XPLUS (BEGIN) - inserted as .XPLUS ()>
;  R0903  1          INDEX$_EMPTYIN,     ! <empty input file '!AS'>
;  R0904  1          INDEX$_IGNORED,     ! <'!AS' ignored>
;  R0905  1          INDEX$_INVINPUT,    ! <invalid input file format in file '!AS'>
;  R0906  1          INDEX$_INVRECORD,   ! <invalid record type in file '!AS'>
;  R0907  1          INDEX$_LASTCONT,    ! <can't generate continuation heading on last page>
;  R0908  1          INDEX$_NOBEGIN,     ! <.XPLUS (END) with no .XPLUS (BEGIN) - inserted as .XPLUS ()>
;  R0909  1          INDEX$_NOEND,       ! <.XPLUS (BEGIN) has no corresponding .XPLUS (END)>
;  R0910  1          INDEX$_NOINDEX,     ! <no index information in file '!AS'>
;  R0911  1          INDEX$_NOLIST,      ! <parameter list not allowed>
;  R0912  1          INDEX$_OVERSTRK,    ! <the following line contains an overstrike sequence>
;  R0913  1          INDEX$_SKIPPED,     ! <!UL reference!%S inside page range - ignored>
;  R0914  1          INDEX$_SYNTAX,      ! <error parsing '!AS'>
;  R0915  1          INDEX$_TEXFILE,     ! <error processing line !UL of TEX character file '!AS'>
;  R0916  1          INDEX$_TOODEEP,     ! <maximum subindex depth exceeded>
;  R0917  1          INDEX$_TOOFEW,      ! <not enough values supplied>
;  R0918  1          INDEX$_TRUNCATED,   ! <string too long - truncated>
;  R0919  1          INDEX$_COMPLETE,    ! <processing complete '!AS'>
;  R0920  1          INDEX$_CREATED,     ! <'!AS' created>
;  R0921  1          INDEX$_IDENT,       ! <INDEX version !AD>
;  R0922  1          INDEX$_PROCFILE,    ! <processing file '!AS'>
;  R0923  1          INDEX$_TEXT,        ! <!AS>
;  R0924  1          INDEX$_TEXTD,       ! <entry text: '!AD'>
;  R0925  1          INDEX$_TMS11;       ! <output file full - continuing with file '!AS'>
;  R0926  1
```

```
;   123     0927  1
;   124     0928  1 %FI
;   125     0929  1
;   126     0930  1 SWITCHES LIST (NOREQUIRE);
;   127     0931  1
;   128     0932  1 !
;   129     0933  1 ! MACROS:
;   130     0934  1 !
;   131     0935  1
;   132     0936  1 MACRO
;   133     0937  1     REPEAT = WHILE 1 DO %;
;   134     0938  1
;   135     0939  1 !
;   136     0940  1 ! EQUATED SYMBOLS:
;   137     0941  1 !
;   138     0942  1
;   139     0943  1 LITERAL
;   140     0944  1     TRUE = 1,
;   141     0945  1     FALSE = 0;
;   142     0946  1
;   143     0947  1 !
;   144     0948  1 ! OWN STORAGE:
;   145     0949  1 !
;   146     0950  1
;   147     0951  1 OWN
;   148     0952  1     CELL        : $C_BLOCK,                         ! Current call characteristics
;   149     0953  1     SORT_STR    : VECTOR [CH$ALLOCATION (1200)],    ! Build sort string here
;   150     0954  1     SORT_PTR,                                       ! Pointer to sort string
;   151     0955  1     SORT_LEN,                                       ! Length of sort string
;   152     0956  1     USER_SORT_LEN,                                  ! Length of user specified sort string
;   153     0957  1     USER_SORT_PTR;                                  ! Pointer to user specified sort string
;   154     0958  1
;   155     0959  1 !
;   156     0960  1 ! EXTERNAL REFERENCES:
;   157     0961  1 !
;   158     0962  1
;   159     0963  1 EXTERNAL LITERAL
;   160     0964  1     TAB     : UNSIGNED (8),             ! TAB character
;   161     0965  1     RINTES  : UNSIGNED (8);             ! Special escape character
;   162     0966  1
;   163     0967  1 EXTERNAL
;   164     0968  1     CMDBLK : $NDXCMD,                   ! Command line information block
;   165     0969  1     XPLBLK : $XPL_BLOCK,                ! Extended indexing information block
;   166     0970  1     BUCKET : $BUCKET_ARRAY [27, 27],    ! Hashing buckets (first character of entry)
;   167     0971  1     BOOKID;                             ! Address of master index book ident string
;   168     0972  1
;   169     0973  1 EXTERNAL ROUTINE
;   170     0974  1     SAVDAT,                             ! Place data in work storage
;   171     0975  1     DMPENT : NOVALUE;
```

```
;  173     0976  1 %SBTTL 'XOUT -- Put away index item'
;  174     0977  1
;  175     0978  1 GLOBAL ROUTINE XOUT (ENTRY_LENGTH, ENTRY_PTR, XTN, BAR_FLAG) : NOVALUE =
;  176     0979  1
;  177     0980  1 !++
;  178     0981  1 ! FUNCTIONAL DESCRIPTION:
;  179     0982  1 !
;  180     0983  1 !       Place an index or sub-index item into the index master storage
;  181     0984  1 !       list in alphabetical order.
;  182     0985  1 !
;  183     0986  1 ! FORMAL PARAMETERS:
;  184     0987  1 !
;  185     0988  1 !       ENTRY_LENGTH    - Length of index entry text
;  186     0989  1 !       ENTRY_PTR       - CH$PTR to index entry text
;  187     0990  1 !       XTN             - Transaction number
;  188     0991  1 !       BAR_FLAG        - Change bar flag
;  189     0992  1 !
;  190     0993  1 ! IMPLICIT INPUTS:
;  191     0994  1 !
;  192     0995  1 !       CMDBLK          - Command line information block
;  193     0996  1 !       XPLBLK          - Extended indexing attributes block
;  194     0997  1 !       CELL            - Information about current position in list
;  195     0998  1 !
;  196     0999  1 ! IMPLICIT OUTPUTS:
;  197     1000  1 !
;  198     1001  1 !       None
;  199     1002  1 !
;  200     1003  1 ! ROUTINE VALUE:
;  201     1004  1 ! COMPLETION CODES:
;  202     1005  1 !
;  203     1006  1 !       NONE
;  204     1007  1 !
;  205     1008  1 ! SIDE EFFECTS:
;  206     1009  1 !
;  207     1010  1 !       Master index is built.
;  208     1011  1 !
;  209     1012  1 !--
;  210     1013  1
;  211     1014  2     BEGIN
;  212     1015  2
;  213     1016  2     LOCAL
;  214     1017  2         INT_HL,
;  215     1018  2         LAST_NB,
;  216     1019  2         STG_PTR,
;  217     1020  2         SUBX_STG,
;  218     1021  2         SUBX_CNT;
;  219     1022  2
;  220     1023  2     !
;  221     1024  2     ! Is this trip necessary?
;  222     1025  2     !
;  223     1026  2     IF .ENTRY_LENGTH EQL 0 THEN RETURN;
;  224     1027  2
;  225     1028  2     !
;  226     1029  2     ! Initialization
;  227     1030  2     !
;  228     1031  3     BEGIN
;  229     1032  3     MAP
```

```
;  230     1033  3          CELL : VECTOR [C$K_LENGTH];
;  231     1034  3
;  232     1035  3      INCR I FROM 0 TO C$K_LENGTH - 1 DO CELL [.I] = 0;
;  233     1036  2      END;
;  234     1037  2
;  235     1038  2      SUBX_STG = .ENTRY_PTR;                      ! Get address of index string
;  236     1039  2      INT_AL = .ENTRY_LENGTH;                     ! Get length of index string.
;  237     1040  2
;  238     1041  2      STG_PTR = .SUBX_STG;
;  239     1042  2      LAST_NB = .SUBX_STG;
;  240     1043  2      SUBX_CNT = 0;
;  241     1044  2
;  242     1045  2      IF .XPLBLK [XPL$V_VALID]
;  243     1046  2      THEN
;  244     1047  2          BEGIN
;  245     1048  2          !
;  246     1049  2          ! Attributes block is valid. Initialize user specified sort parameters.
;  247     1050  2          !
;  248     1051  2          BIND
;  249     1052  2              SORT_STR = XPLBLK [XPL$T_SORT] : $STR_DESCRIPTOR ();
;  250     1053  2
;  251     1054  2          USER_SORT_LEN = .SORT_STR [STR$H_LENGTH];
;  252     1055  2          USER_SORT_PTR = .SORT_STR [STR$A_POINTER];
;  253     1056  2          END
;  254     1057  2      ELSE
;  255     1058  2          USER_SORT_LEN = 0;
;  256     1059  2
;  257     1060  2      !
;  258     1061  2      ! Scan the entire character string
;  259     1062  2      !
;  260     1063  2      INCR I FROM 1 TO .INT_HL DO
;  261     1064  3          BEGIN
;  262     1065  3
;  263     1066  3          LOCAL
;  264     1067  3              CHARACTER;
;  265     1068  3
;  266     1069  3          CHARACTER = CH$RCHAR_A (STG_PTR);
;  267     1070  3
;  268     1071  3          !
;  269     1072  3          ! Look for special handling
;  270     1073  3          !
;  271     1074  3          IF .CHARACTER EQL RINTES
;  272     1075  3          THEN
;  273     1076  4              BEGIN
;  274     1077  4              !
;  275     1078  4              ! Interpret escape sequences.
;  276     1079  4              !
;  277     1080  4              CHARACTER = CH$RCHAR_A (STG_PTR);
;  278     1081  4              I = .I + 1;
;  279     1082  4
;  280     1083  4              IF .CHARACTER EQL %C'J'
;  281     1084  4              THEN
;  282     1085  5                  BEGIN
;  283     1086  5                  !
;  284     1087  5                  ! Set up sub-index
;  285     1088  5                  !
;  286     1089  5                  LOCAL
```

```
;  287     1090  5                    T_PTR : REF $XE_BLOCK;
;  288     1091  5
;  289     1092  5                 CH$RCHAR_A (STG_PTR);                   ! Skip null argument
;  290     1093  5                 I = .I + 1;
;  291     1094  5
;  292     1095  5                 !
;  293     1096  5                 ! Set up sort string
;  294     1097  5                 !
;  295     1098  5                 SORT_AS (.SUBX_STG, CH$DIFF (.LAST_NB, .SUBX_STG), .SUBX_CNT);
;  296     1099  5
;  297     1100  5                 !
;  298     1101  5                 ! Look for entry
;  299     1102  5                 !
;  300     1103  5                 FIND_POS (.SUBX_STG, CH$DIFF (.LAST_NB, .SUBX_STG), .SUBX_CNT, FALSE, 0);
;  301     1104  5
;  302     1105  5                 !
;  303     1106  5                 ! Enter it if it is not already there
;  304     1107  5                 !
;  305     1108  5                 IF NOT .CELL[C$V_IDNS]
;  306     1109  5                 THEN
;  307     1110  5                     INSERT_INX (.SUBX_STG, CH$DIFF (.LAST_NB, .SUBX_STG), .SUBX_CNT, 0, .BAR_FLAG, .ENTRY_LE
;  308     1111  5
;  309     1112  5                 !
;  310     1113  5                 ! Clear out sort string
;  311     1114  5                 !
;  312     1115  5                 SORT_LEN = 0;
;  313     1116  5                 SORT_PTR = 0;
;  314     1117  5
;  315     1118  5                 !
;  316     1119  5                 ! Skip over text
;  317     1120  5                 !
;  318     1121  5                 SUBX_STG = .STG_PTR;
;  319     1122  5                 LAST_NB = .SUBX_STG;
;  320     1123  5                 SUBX_CNT = .SUBX_CNT + 1;
;  321     1124  5                 CELL [C$V_IDNS] = FALSE;
;  322     1125  5
;  323     1126  5                 !
;  324     1127  5                 ! Is there a sub-index list?
;  325     1128  5                 !
;  326     1129  5                 T_PTR = .CELL [C$A_CURR];
;  327     1130  5                 IF .T_PTR [XE$A_SUBX] EQL 0
;  328     1131  5                 THEN
;  329     1132  5                     !
;  330     1133  5                     ! Insert end of sub-index list
;  331     1134  5                     !
;  332     1135  5                     INSERT_INX (0, 0, .SUBX_CNT, 0, .BAR_FLAG, .ENTRY_LENGTH, .ENTRY_PTR)
;  333     1136  5                 ELSE
;  334     1137  6                     BEGIN
;  335     1138  6                     !
;  336     1139  6                     ! Set pointer to head of sub list
;  337     1140  6                     !
;  338     1141  6                     CELL [C$A_PREV] = .CELL [C$A_CURR];
;  339     1142  6                     CELL [C$A_CURR] = .T_PTR [XE$A_SUBX]
;  340     1143  6                     END
;  341     1144  5                 END
;  342     1145  4             ELSE
;  343     1146  4                 LAST_NB = .STG_PTR
```

```
;  344     1147  4              END
;  345     1148  3          ELSE
;  346     1149  3              IF .CHARACTER NEQ %C' ' THEN LAST_NB = .STG_PTR
;  347     1150  3
;  348     1151  3          END;
;  349     1152  3
;  350     1153  3      !
;  351     1154  3      ! End of line was reached
;  352     1155  3      !
;  353     1156  2      SORT_AS (.SUBX_STG, CH$DIFF (.LAST_NB, .SUBX_STG), .SUBX_CNT);
;  354     1157  2      FIND_POS (.SUBX_STG, CH$DIFF (.LAST_NB, .SUBX_STG), .SUBX_CNT, TRUE, .XTN);
;  355     1158  2      INSERT_INX (.SUBX_STG, CH$DIFF (.LAST_NB, .SUBX_STG), .SUBX_CNT, .XTN, .BAR_FLAG, .ENTRY_LENGTH, .ENTRY_
;  356     1159  2
;  357     1160  1      END;                            !End of XOUT

                                                        .TITLE  NDXOUT NDXOUT -- Sort and store index entries
                                                        .IDENT  \V04-000\

                                                        .PSECT  $OWN$,NOEXE,2

                                                00000 CELL:     .BLKB   16
                                                00010 SORT_STR:
                                                                .BLKB   1200
                                                004C0 SORT_PTR:
                                                                .BLKB   4
                                                004C4 SORT_LEN:
                                                                .BLKB   4
                                                004C8 USER_SORT_LEN:
                                                                .BLKB   4
                                                004CC USER_SORT_PTR:
                                                                .BLKB   4

                                                        .EXTRN  DSRINDEX$_BADLOGIC
                                                        .EXTRN  DSRINDEX$_BADVALUE
                                                        .EXTRN  DSRINDEX$_INSVIRMEM
                                                        .EXTRN  DSRINDEX$_LINELENG
                                                        .EXTRN  DSRINDEX$_NOREF
                                                        .EXTRN  DSRINDEX$_OPENIN
                                                        .EXTRN  DSRINDEX$_OPENOUT
                                                        .EXTRN  DSRINDEX$_TOOMANY
                                                        .EXTRN  DSRINDEX$_VALERR
                                                        .EXTRN  DSRINDEX$_CANTBAL
                                                        .EXTRN  DSRINDEX$_CLOSEQUOT
                                                        .EXTRN  DSRINDEX$_CONFQUAL
                                                        .EXTRN  DSRINDEX$_CTRLCHAR
                                                        .EXTRN  DSRINDEX$_DOESNTFIT
                                                        .EXTRN  DSRINDEX$_DUPBEGIN
                                                        .EXTRN  DSRINDEX$_EMPTYIN
                                                        .EXTRN  DSRINDEX$_IGNORED
                                                        .EXTRN  DSRINDEX$_INVINPUT
                                                        .EXTRN  DSRINDEX$_INVRECORD
                                                        .EXTRN  DSRINDEX$_LASTCONT
                                                        .EXTRN  DSRINDEX$_NOBEGIN
                                                        .EXTRN  DSRINDEX$_NOEND
                                                        .EXTRN  DSRINDEX$_NOINDEX
                                                        .EXTRN  DSRINDEX$_NOLIST
```

```
                                                        .EXTRN  DSRINDEX$_OVERSTRK
                                                        .EXTRN  DSRINDEX$_SKIPPED
                                                        .EXTRN  DSRINDEX$_SYNTAX
                                                        .EXTRN  DSRINDEX$_TEXFILE
                                                        .EXTRN  DSRINDEX$_TOODEEP
                                                        .EXTRN  DSRINDEX$_TOOFEW
                                                        .EXTRN  DSRINDEX$_TRUNCATED
                                                        .EXTRN  DSRINDEX$_COMPLETE
                                                        .EXTRN  DSRINDEX$_CREATED
                                                        .EXTRN  DSRINDEX$_IDENT
                                                        .EXTRN  DSRINDEX$_PROCFILE
                                                        .EXTRN  DSRINDEX$_TEXT, DSRINDEX$_TEXTD
                                                        .EXTRN  DSRINDEX$_TMS11
                                                        .EXTRN  TAB, RINTES, CMDBLK
                                                        .EXTRN  XPLBLK, BUCKET, BOOKID
                                                        .EXTRN  SAVDAT, DMPENT

                                                        .PSECT  $CODE$,NOWRT,2

                                    0FFC 00000          .ENTRY  XOUT, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11  ; 0978
                5B 00000000G   EF    9E 00002          MOVAB   XPLBLK, R11
                5A 00000000'   EF    9E 00009          MOVAB   CELL, R10
                56       04    AC    D0 00010          MOVL    ENTRY_LENGTH, R6                            ; 1026
                               01    12 00014          BNEQ    1$
                                     04 00016          RET
                               50    D4 00017 1$:      CLRL    I                                           ; 1035
                             6A40    D4 00019 2$:      CLRL    CELL[I]
                   F9          50    03    F3 0001C    AOBLEQ  #3, I, 2$
                               54    08    AC    D0 00020    MOVL    ENTRY_PTR, SUBX_STG                   ; 1038
                               59          56    D0 00024    MOVL    R6, INT_HL                            ; 1039
                               57          54    D0 00027    MOVL    SUBX_STG, STG_PTR                     ; 1041
                               52          54    D0 0002A    MOVL    SUBX_STG, LAST_NB                     ; 1042
                                           55    D4 0002D    CLRL    SUBX_CNT                              ; 1043
                               0E          6B    E9 0002F    BLBC    XPLBLK, 3$                            ; 1045
                     04C8      CA    04    AB    3C 00032    MOVZWL  SORT_STR, USER_SORT_LEN               ; 1054
                     04CC      CA    08    AB    D0 00038    MOVL    SORT_STR+4, USER_SORT_PTR             ; 1055
                                           04    11 0003E    BRB     4$                                    ; 1045
                                 04C8      CA    D4 00040 3$:      CLRL    USER_SORT_LEN                 ; 1058
                                           58    D4 00044 4$:      CLRL    I                             ; 1063
                                       0099    31 00046          BRW     11$
                               50          87    9A 00049 5$:      MOVZBL  (STG_PTR)+, CHARACTER         ; 1069
                  00000000G    8F          50    D1 0004C    CMPL    CHARACTER, #RINTES                  ; 1074
                                           03    13 00053    BEQL    6$
                                       0082    31 00055          BRW     9$
                               50          87    9A 00058 6$:      MOVZBL  (STG_PTR)+, CHARACTER         ; 1080
                                           58    D6 0005B    INCL    I                                   ; 1081
                  0000004A     8F          50    D1 0005D    CMPL    CHARACTER, #74                      ; 1083
                                           79    12 00064    BNEQ    10$
                                           57    D6 00066    INCL    STG_PTR                             ; 1092
                                           58    D6 00068    INCL    I                                   ; 1093
                                           55    DD 0006A    PUSHL   SUBX_CNT                            ; 1098
                   53          52          54    C3 0006C    SUBL3   SUBX_STG, LAST_NB, R3
                                           53    DD 00070    PUSHL   R3
                                           54    DD 00072    PUSHL   SUBX_STG
                  00000000V    EF          03    FB 00074    CALLS   #3, SORT_AS
                                           7E    7C 0007B    CLRQ    -(SP)                               ; 1103
                                           28    BB 0007D    PUSHR   #^M<R3,R5>
```

```
                                                54 DD 0007F          PUSHL   SUBX_STG
                                                05 FB 00081          CALLS   #5, FIND_POS
                00000000V EF                                                                    :
                          15                    AA E8 00088          BLBS    CELL+12, 7$       : 1108
                                          0C    AC DD 0008C          PUSHL   ENTRY_PTR         : 1110
                                          08    56 DD 0008F          PUSHL   R6                :
                                                AC DD 00091          PUSHL   BAR_FLAG          :
                                          10    7E D4 00094          CLRL    -(SP)             :
                                                28 BB 00096          PUSHR   #^M<R3,R5>        :
                                                54 DD 00098          PUSHL   SUBX_STG          :
                00000000V EF                    07 FB 0009A          CALLS   #7, INSERT_INX    :
                                          04C0  CA 7C 000A1 7$:      CLRQ    SORT_PTR          : 1116
                          54                    57 D0 000A5          MOVL    STG_PTR, SUBX_STG : 1121
                          52                    54 D0 000A8          MOVL    SUBX_STG, LAST_NB : 1122
                                                55 D6 000AB          INCL    SUBX_CNT          : 1123
                     0C   AA                    01 8A 000AD          BICB2   #1, CELL+12       : 1124
                          50                    6A D0 000B1          MOVL    CELL, T_PTR       : 1129
                                          08    A0 D5 000B4          TSTL    8(T_PTR)          : 1130
                                                17 12 000B7          BNEQ    8$                :
                                          08    AC DD 000B9          PUSHL   ENTRY_PTR         : 1135
                                                56 DD 000BC          PUSHL   R6                :
                                          10    AC DD 000BE          PUSHL   BAR_FLAG          :
                                                7E D4 000C1          CLRL    -(SP)             :
                                                55 DD 000C3          PUSHL   SUBX_CNT          :
                                                7E 7C 000C5          CLRQ    -(SP)             :
                00000000V EF                    07 FB 000C7          CALLS   #7, INSERT_INX    :
                                                12 11 000CE          BRB     11$               :
                     04   AA                    6A D0 000D0 8$:      MOVL    CELL, CELL+4      : 1141
                          6A                08  A0 D0 000D4          MOVL    8(T_PTR), CELL    : 1142
                                                08 11 000D8          BRB     11$               : 1130
                          20                    50 D1 000DA 9$:      CMPL    CHARACTER, #32    : 1149
                                                03 13 000DD          BEQL    11$               :
                          52                    57 D0 000DF 10$:     MOVL    STG_PTR, LAST_NB  :
                FF61      58                  01 59 F1 000E2 11$:    ACBL    INT_HL, #1, I, 5$ : 1074
                                                55 DD 000E8          PUSHL   SUBX_CNT          : 1156
                          52                    54 C2 000EA          SUBL2   SUBX_STG, R2      :
                                                52 DD 000ED          PUSHL   R2                :
                                                54 DD 000EF          PUSHL   SUBX_STG          :
                     00000000V EF                03 FB 000F1          CALLS   #3, SORT_AS       :
                                          0C    AC DD 000F8          PUSHL   XTN               : 1157
                                                01 DD 000FB          PUSHL   #1                :
                                                24 BB 000FD          PUSHR   #^M<R2,R5>        :
                                                54 DD 000FF          PUSHL   SUBX_STG          :
                00000000V EF                    05 FB 00101          CALLS   #5, FIND_POS      :
                                          08    AC DD 00108          PUSHL   ENTRY_PTR         : 1158
                                                56 DD 0010B          PUSHL   R6                :
                          7E              0C    AC 7D 0010D          MOVQ    XTN, -(SP)        :
                                                24 BB 00111          PUSHR   #^M<R2,R5>        :
                                                54 DD 00113          PUSHL   SUBX_STG          :
                00000000V EF                    07 FB 00115          CALLS   #7, INSERT_INX    :
                                                04 0011C             RET                       : 1160

; Routine Size:  285 bytes,    Routine Base:  $CODE$ + 0000
```

```
;   359          1161  1 %SBTTL 'SORT_AS -- Build sort string'
;   360          1162  1 ROUTINE SORT_AS (I_PTR, I_LEN, LEVEL) : NOVALUE =
;   361          1163  1 !++
;   362          1164  1 !
;   363          1165  1 ! FUNCTIONAL DESCRIPTION:
;   364          1166  1 !
;   365          1167  1 !       This routine builds the sort string used to position the index
;   366          1168  1 !       entry in the index.
;   367          1169  1 !
;   368          1170  1 !       If the user specified a sort string then that string is used.
;   369          1171  1 !
;   370          1172  1 !       If /SORT=LETTER was specified on the command line, a sort string
;   371          1173  1 !       is built from the input string.
;   372          1174  1 !
;   373          1175  1 !       If /SORT=NONALPHA=IGNORE was specified on the command line, a
;   374          1176  1 !       sort string is built if the first character in the input string
;   375          1177  1 !       is not alphabetic.
;   376          1178  1 !
;   377          1179  1 !       Otherwise no string is built and the entry is positioned
;   378          1180  1 !       according to the text of the entry.
;   379          1181  1 !
;   380          1182  1 ! FORMAL PARAMETERS:
;   381          1183  1 !
;   382          1184  1 !       I_LEN   - Length of input string
;   383          1185  1 !       I_PTR   - Pointer to input string
;   384          1186  1 !       LEVEL   - Subindex level
;   385          1187  1 !
;   386          1188  1 ! IMPLICIT INPUTS:
;   387          1189  1 !
;   388          1190  1 !       XPLBLK          - Extended index attributes block
;   389          1191  1 !       CMDBLK          - Command line information block
;   390          1192  1 !       USER_SORT_LEN   - Length of user specified sort string if any
;   391          1193  1 !       USER_SORT_PTR   - Pointer to user specified sort string if any
;   392          1194  1 !
;   393          1195  1 ! IMPLICIT OUTPUTS:
;   394          1196  1 !
;   395          1197  1 !       USER_SORT_LEN   - Length of remainder of user specified sort string if any
;   396          1198  1 !       USER_SORT_PTR   - Pointer to remainder of user specified sort string if any
;   397          1199  1 !       SORT_PTR        - Points to the sort string if any
;   398          1200  1 !       SORT_LEN        - Is the length of the sort string
;   399          1201  1 !
;   400          1202  1 ! ROUTINE VALUE:
;   401          1203  1 ! COMPLETION CODES:
;   402          1204  1 !
;   403          1205  1 !       None
;   404          1206  1 !
;   405          1207  1 ! SIDE EFFECTS:
;   406          1208  1 !
;   407          1209  1 !       None
;   408          1210  1 !--
;   409          1211  2     BEGIN
;   410          1212  2
;   411          1213  2     LOCAL
;   412          1214  2         LEN,
;   413          1215  2         PTR;
;   414          1216  2
;   415          1217  2     SORT_LEN = 0;
```

```
;  416     1218  2     SORT_PTR = CH$PTR (SORT_STR);
;  417     1219  2
;  418     1220  2     IF .USER_SORT_LEN NEQ 0
;  419     1221  2     THEN
;  420     1222  3         BEGIN
;  421     1223  3         !
;  422     1224  3         ! User specified a sort string.
;  423     1225  3         ! Get the next segment.
;  424     1226  3         !
;  425     1227  3         SORT_PTR = .USER_SORT_PTR;
;  426     1228  3
;  427     1229  3         WHILE .USER_SORT_LEN GTR 0 DO
;  428     1230  4             BEGIN
;  429     1231  4
;  430     1232  4             LOCAL
;  431     1233  4                 CH;
;  432     1234  4
;  433     1235  4             CH = CH$RCHAR_A (USER_SORT_PTR);                ! Get next character
;  434     1236  4             USER_SORT_LEN = .USER_SORT_LEN - 1;             ! One less character in string
;  435     1237  4
;  436     1238  4             IF .CH EQL RINTES
;  437     1239  4             THEN
;  438     1240  5                 BEGIN                                       ! RUNOFF escape sequence
;  439     1241  5
;  440     1242  5                 CH = CH$RCHAR_A (USER_SORT_PTR);
;  441     1243  5                 CH$RCHAR_A (USER_SORT_PTR);
;  442     1244  5                 USER_SORT_LEN = .USER_SORT_LEN - 2;
;  443     1245  5
;  444     1246  5                 IF .CH EQL %C'J' THEN EXITLOOP;             ! Subindex sequence signals end of string
;  445     1247  5
;  446     1248  5                 SORT_LEN = .SORT_LEN + 3;                   ! 3 more characters in sort string
;  447     1249  5                 END
;  448     1250  4             ELSE
;  449     1251  4                 SORT_LEN = .SORT_LEN + 1;                   ! 1 more character in sort string
;  450     1252  4
;  451     1253  3             END;
;  452     1254  3
;  453     1255  3         RETURN;
;  454     1256  2         END;
;  455     1257  2
;  456     1258  2     LEN = .I_LEN;
;  457     1259  2     PTR = .I_PTR;
;  458     1260  2
;  459     1261  2     SELECTONE .CMDBLK [NDX$H_NONALPHA] OF
;  460     1262  2     SET
;  461     1263  2
;  462     1264  2     [IGNORE]:
;  463     1265  3         BEGIN
;  464     1266  3         !
;  465     1267  3         ! Ignore leading non-alphas
;  466     1268  3         !
;  467     1269  3         LOCAL
;  468     1270  3             SCAN_PTR,
;  469     1271  3             FIRST_PTR,
;  470     1272  3             FIRST_LEN;
;  471     1273  3
;  472     1274  3         FIRST_PTR = .PTR;
```

```
;  473     1275  3          FIRST_LEN = 0;
;  474     1276  3          SCAN_PTR = .PTR;
;  475     1277  3
;  476     1278  3          DECR I FROM .LEN TO 1 DO
;  477     1279  4              BEGIN
;  478     1280  4              LOCAL
;  479     1281  4                  CH;
;  480     1282  4
;  481     1283  4              CH = CH$RCHAR_A (SCAN_PTR);
;  482     1284  4
;  483     1285  4              IF .CH EQL RINTES
;  484     1286  4              THEN
;  485     1287  5                  BEGIN
;  486     1288  5                  !
;  487     1289  5                  ! RUNOFF escape sequence
;  488     1290  5                  !
;  489     1291  5                  IF .FIRST_LEN EQL 0
;  490     1292  5                  THEN
;  491     1293  6                      BEGIN
;  492     1294  6                      !
;  493     1295  6                      ! Save pointer and length if first escape sequence seen
;  494     1296  6                      !
;  495     1297  6                      FIRST_LEN = .I;
;  496     1298  6                      FIRST_PTR = CH$PLUS (.SCAN_PTR, -1);
;  497     1299  5                      END;
;  498     1300  5
;  499     1301  5                  CH$RCHAR_A (SCAN_PTR);          ! Skip rest of sequence
;  500     1302  5                  CH$RCHAR_A (SCAN_PTR);
;  501     1303  5                  I = .I - 2;                     ! Decrement length remaining
;  502     1304  5                  END
;  503     1305  4              ELSE
;  504     1306  5                  BEGIN
;  505     1307  6                  IF LETTER (.CH)
;  506     1308  5                  THEN
;  507     1309  6                      BEGIN
;  508     1310  6                      !
;  509     1311  6                      ! Alphabetic character
;  510     1312  6                      !
;  511     1313  6                      IF .FIRST_LEN EQL 0
;  512     1314  6                      THEN
;  513     1315  7                          BEGIN
;  514     1316  7                          !
;  515     1317  7                          ! No RUNOFF escape sequence was seen.
;  516     1318  7                          ! Save pointer and length.
;  517     1319  7                          !
;  518     1320  7                          FIRST_LEN = .I;
;  519     1321  7                          FIRST_PTR = CH$PLUS (.SCAN_PTR, -1);
;  520     1322  6                          END;
;  521     1323  6
;  522     1324  6                      EXITLOOP;
;  523     1325  6                      END
;  524     1326  5                  ELSE
;  525     1327  5                      FIRST_LEN = 0;
;  526     1328  4                  END;
;  527     1329  3              END;
;  528     1330  3
;  529     1331  3          IF .FIRST_LEN NEQ 0
```

```
;   530         1332  3          THEN
;   531         1333  4              BEGIN
;   532         1334  4              !
;   533         1335  4              ! Found an alphabetic sequence
;   534         1336  4              !
;   535         1337  4              LEN = .FIRST_LEN;
;   536         1338  4              PTR = .FIRST_PTR;
;   537         1339  3              END;
;   538         1340  2          END;
;   539         1341  2
;   540         1342  2      [AFTER]:
;   541         1343  3          BEGIN
;   542         1344  3          !
;   543         1345  3          ! Put leading nonalphas after
;   544         1346  3          !
;   545         1347  3          IF .LEVEL NEQ 0
;   546         1348  3          THEN
;   547         1349  4              BEGIN
;   548         1350  4              !
;   549         1351  4              ! Build a sort string for all but top level entries.
;   550         1352  4              ! Top level entries are sorted after by examining the
;   551         1353  4              ! nonalpha bucket last.
;   552         1354  4              !
;   553         1355  4              LOCAL
;   554         1356  4                  SCAN_PTR;
;   555         1357  4
;   556         1358  4              SCAN_PTR = .PTR;
;   557         1359  4
;   558         1360  4              INCR I FROM 1 TO .LEN DO
;   559         1361  5                  BEGIN
;   560         1362  5                  LOCAL
;   561         1363  5                      CH;
;   562         1364  5
;   563         1365  5                  CH = CH$RCHAR_A (SCAN_PTR);
;   564         1366  5
;   565         1367  5                  IF .CH EQL RINTES
;   566         1368  5                  THEN
;   567         1369  6                      BEGIN
;   568         1370  6                      !
;   569         1371  6                      ! RUNOFF escape sequence - skip over it
;   570         1372  6                      !
;   571         1373  6                      CH$RCHAR_A (SCAN_PTR);
;   572         1374  6                      CH$RCHAR_A (SCAN_PTR);
;   573         1375  6                      I = .I + 2;
;   574         1376  6                      END
;   575         1377  5                  ELSE
;   576         1378  6                      BEGIN
;   577         1379  6                      !
;   578         1380  6                      ! Have first character
;   579         1381  6                      !
;   580         1382  7                      IF NOT LETTER (.CH)
;   581         1383  6                      THEN
;   582         1384  7                          BEGIN
;   583         1385  7                          !
;   584         1386  7                          ! Leading nonalpha.
;   585         1387  7                          ! Make it sort after by building a sort string
;   586         1388  7                          ! which starts with 'zzzz'
```

```
;   587     1389  7                         !
;   588     1390  7                         CH$COPY (4, CH$PTR (UPLIT ('zzzz')), .LEN, .PTR, %C' ', .LEN + 4, CH$PTR (SORT_STR))
;   589     1391  7
;   590     1392  7                         LEN = .LEN + 4;
;   591     1393  7                         PTR = CH$PTR (SORT_STR);
;   592     1394  6                         END;
;   593     1395  6
;   594     1396  6                     EXITLOOP;
;   595     1397  5                     END;
;   596     1398  4                 END;
;   597     1399  3             END;
;   598     1400  2         END;
;   599     1401  2
;   600     1402  2         [BEFORE]:
;   601     1403  2             !
;   602     1404  2             ! Sort nonalphas before.
;   603     1405  2             ! Since this is the default, no action is required.
;   604     1406  2             !
;   605     1407  2             ;
;   606     1408  2
;   607     1409  2         TES;
;   608     1410  2
;   609     1411  2         IF .LEN NEQ .I_LEN
;   610     1412  2         THEN
;   611     1413  3             BEGIN
;   612     1414  3             !
;   613     1415  3             ! A sort string has been built.
;   614     1416  3             ! Save pointer and length of resulting string
;   615     1417  3             !
;   616     1418  3             SORT_LEN = .LEN;
;   617     1419  3             SORT_PTR = .PTR;
;   618     1420  2             END;
;   619     1421  2
;   620     1422  2         IF NOT .CMDBLK [NDX$V_WORD_SORT]
;   621     1423  2         THEN
;   622     1424  3             BEGIN
;   623     1425  3             !
;   624     1426  3             ! Letter by letter sort - remove whitespace.
;   625     1427  3             !
;   626     1428  3             LOCAL
;   627     1429  3                 RINTES_PTR,
;   628     1430  3                 RINTES_LEN,
;   629     1431  3                 SCAN_PTR;
;   630     1432  3
;   631     1433  3             RINTES_PTR = 0;
;   632     1434  3             RINTES_LEN = 0;
;   633     1435  3             SCAN_PTR = .PTR;
;   634     1436  3
;   635     1437  3             SORT_PTR = CH$PTR (SORT_STR);
;   636     1438  3             SORT_LEN = 0;
;   637     1439  3
;   638     1440  3             INCR I FROM 1 TO .LEN DO
;   639     1441  4                 BEGIN
;   640     1442  4                 LOCAL
;   641     1443  4                     CH;
;   642     1444  4
;   643     1445  4                 CH = CH$RCHAR_A (SCAN_PTR);
```

```
;  644     1446  4
;  645     1447  4                IF .CH EQL RINTES
;  646     1448  4                THEN
;  647     1449  5                    BEGIN
;  648     1450  5                    !
;  649     1451  5                    ! RUNOFF escape sequence.
;  650     1452  5                    !
;  651     1453  5                    IF .RINTES_LEN EQL 0
;  652     1454  5                    THEN
;  653     1455  6                        BEGIN
;  654     1456  6                        !
;  655     1457  6                        ! Not a multiple sequence.
;  656     1458  6                        ! Save pointer to beginning of output sequence.
;  657     1459  6                        !
;  658     1460  6                        RINTES_LEN = .SORT_LEN;
;  659     1461  6                        RINTES_PTR = .SORT_PTR;
;  660     1462  5                        END;
;  661     1463  5
;  662     1464  5                    CH$WCHAR_A (.CH, SORT_PTR);
;  663     1465  5                    CH$WCHAR_A (CH$RCHAR_A (SCAN_PTR), SORT_PTR);
;  664     1466  5                    CH$WCHAR_A (CH$RCHAR_A (SCAN_PTR), SORT_PTR);
;  665     1467  5                    SORT_LEN = .SORT_LEN + 3;
;  666     1468  5                    I = .I + 2;
;  667     1469  5                    END
;  668     1470  4                ELSE
;  669     1471  5                    BEGIN
;  670     1472  6                    IF (.CH EQL %C' ') OR (.CH EQL TAB) OR (.CH EQL %C'-')
;  671     1473  5                    THEN
;  672     1474  6                        BEGIN
;  673     1475  6                        !
;  674     1476  6                        ! Whitespace.
;  675     1477  6                        !
;  676     1478  6                        IF .RINTES_PTR NEQ 0
;  677     1479  6                        THEN
;  678     1480  7                            BEGIN
;  679     1481  7                            !
;  680     1482  7                            ! Whitespace was emphasized.
;  681     1483  7                            ! Remove emphasis from output string
;  682     1484  7                            !
;  683     1485  7                            SORT_PTR = .RINTES_PTR;
;  684     1486  7                            SORT_LEN = .RINTES_LEN;
;  685     1487  7
;  686     1488  7                            RINTES_PTR = 0;
;  687     1489  7                            RINTES_LEN = 0;
;  688     1490  6                            END;
;  689     1491  6                        END
;  690     1492  5                    ELSE
;  691     1493  6                        BEGIN
;  692     1494  6                        !
;  693     1495  6                        ! Some other character
;  694     1496  6                        !
;  695     1497  6                        CH$WCHAR_A (.CH, SORT_PTR);
;  696     1498  6                        SORT_LEN = .SORT_LEN + 1;
;  697     1499  6
;  698     1500  6                        RINTES_PTR = 0;
;  699     1501  6                        RINTES_LEN = 0;
;  700     1502  5                        END;
```

```
;  701          1503  4                 END;
;  702          1504  4             END;
;  703          1505  4
;  704          1506  4         SORT_PTR = CH$PTR (SORT_STR);
;  705          1507  4         END;
;  706          1508  4
;  707          1509  1     END;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                                7A  7A  7A  7A  00000 P.AAA:    .ASCII  \zzzz\                                          ;


                                                        .PSECT  $CODE$,NOWRT,2

                                          0FFC 00000 SORT_AS:.WORD  Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11                  ; 1162
                            5E                  04   C2 00002       SUBL2   #4, SP
                                00000000'       EF   D4 00005       CLRL    SORT_LEN                                    ; 1217
                00000000'   EF  00000000'       EF   9E 0000B       MOVAB   SORT_STR, SORT_PTR                          ; 1218
                                00000000'       EF   D5 00016       TSTL    USER_SORT_LEN                               ; 1220
                                                65   13 0001C       BEQL    5$
                00000000'   EF  00000000'       EF   D0 0001E       MOVL    USER_SORT_PTR, SORT_PTR                     ; 1227
                                0000000U'       EF   D5 00029 1$:   TSTL    USER_SORT_LEN                               ; 1229
                                                01   14 0002F       BGTR    2$
                                                     04 00031       RET
                            50  00000000'       FF   9A 00032 2$:   MOVZBL  @USER_SORT_PTR, CH                          ; 1235
                                00000000'       EF   D6 00039       INCL    USER_SORT_PTR
                                00000000'       EF   D7 0003F       DECL    USER_SORT_LEN                               ; 1236
                00000000G   8F                  50   D1 00045       CMPL    CH, #RINTES                                 ; 1238
                                                2D   12 0004C       BNEQ    4$
                            50  00000000'       FF   9A 0004E       MOVZBL  @USER_SORT_PTR, CH                          ; 1242
                                00000000'       EF   D6 00055       INCL    USER_SORT_PTR
                                00000000'       EF   D6 0005B       INCL    USER_SORT_PTR                               ; 1243
                00000000'   EF                  02   C2 00061       SUBL2   #2, USER_SORT_LEN                           ; 1244
                0000004A    8F                  50   D1 00068       CMPL    CH, #74                                     ; 1246
                                                01   12 0006F       BNEQ    3$
                                                     04 00071       RET
                00000000'   EF                  03   C0 00072 3$:   ADDL2   #3, SORT_LEN                               ; 1248
                                                AE   11 00079       BRB     1$                                          ; 1238
                                00000000'       EF   D6 00078 4$:   INCL    SORT_LEN                                    ; 1251
                                                A6   11 00081       BRB     1$                                          ; 1229
                            56          08      AC   D0 00083 5$:   MOVL    I_LEN, LEN                                  ; 1258
                            57          04      AC   D0 00087       MOVL    I_PTR, PTR                                  ; 1259
                            50  00000000G       EF   32 0008B       CVTWL   CMDBLK+8, R0                                ; 1261
                            03                  50   B1 00092       CMPW    R0, #3                                      ; 1264
                                                6F   12 00095       BNEQ    14$
                            51                  57   D0 00097       MOVL    PTR, FIRST_PTR                              ; 1274
                                                54   D4 0009A       CLRL    FIRST_LEN                                   ; 1275
                            52                  57   D0 0009C       MOVL    PTR, SCAN_PTR                               ; 1276
                            50          01      A6   9E 0009F       MOVAB   1(R6), I                                    ; 1278
                                                52   11 000A3       BRB     12$
                            53                  82   9A 000A5 6$:   MOVZBL  (SCAN_PTR)+, CH                            ; 1283
                00000000G   8F                  53   D1 000A8       CMPL    CH, #RINTES                                 ; 1285
                                                13   12 000AF       BNEQ    8$
```

```
                                 54  D5 000B1          TSTL    FIRST_LEN                                   ; 1291
                                 07  12 000B3          BNEQ    7$
                              54 50  D0 000B5          MOVL    I, FIRST_LEN                                ; 1297
                              51 FF  A2 9E 000B8       MOVAB   -1(R2), FIRST_PTR                           ; 1298
                              52 02  C0 000BC 7$:      ADDL2   #2, SCAN_PTR                               ; 1302
                              50 02  C2 000BF          SUBL2   #2, I                                       ; 1303
                                 33  11 000C2          BRB     12$                                         ; 1285
                  00000041    8F     53  D1 000C4 8$:  CMPL    CH, #65                                     ; 1307
                                 09  19 000CB          BLSS    9$
                  0000005A    8F     53  D1 000CD      CMPL    CH, #90
                                 12  15 000D4          BLEQ    10$
                  00000061    8F     53  D1 000D6 9$:  CMPL    CH, #97
                                 16  19 000DD          BLSS    11$
                  0000007A    8F     53  D1 000DF      CMPL    CH, #122
                                 0D  14 000E6          BGTR    11$
                                 54  D5 000E8 10$:     TSTL    FIRST_LEN                                   ; 1313
                                 0E  12 000EA          BNEQ    13$
                              54 50  D0 000EC          MOVL    I, FIRST_LEN                                ; 1320
                              51 FF  A2 9E 000EF       MOVAB   -1(R2), FIRST_PTR                           ; 1321
                                 05  11 000F3          BRB     13$                                         ; 1309
                                 54  D4 000F5 11$:     CLRL    FIRST_LEN                                   ; 1327
                              AB     50  F5 000F7 12$:  SOBGTR  I, 6$                                       ; 1278
                                 54  D5 000FA 13$:     TSTL    FIRST_LEN                                   ; 1331
                                 56  13 000FC          BEQL    18$
                              56 54  D0 000FE          MOVL    FIRST_LEN, LEN                              ; 1337
                              57 51  D0 00101          MOVL    FIRST_PTR, PTR                              ; 1338
                                 7D  11 00104          BRB     21$                                         ; 1261
                              02     50  B1 00106 14$:  CMPW    R0, #2                                      ; 1342
                                 7E  12 00109          BNEQ    23$
                           0C    AC  D5 0010B          TSTL    LEVEL                                       ; 1347
                                 79  13 0010E          BEQL    23$
                              6E 57  D0 00110          MOVL    PTR, SCAN_PTR                               ; 1358
                              5B 56  D0 00113          MOVL    LEN, R11                                    ; 1360
                                 5A  D4 00116          CLRL    I
                                 6B  11 00118          BRB     22$
                           50    00  BE 9A 0011A 15$:  MOVZBL  @SCAN_PTR, CH                               ; 1365
                                 6E  D6 0011E          INCL    SCAN_PTR
                  00000000G    8F     50  D1 00120      CMPL    CH, #RINTES                                 ; 1367
                                 09  12 00127          BNEQ    16$
                                 6E  D6 00129          INCL    SCAN_PTR                                    ; 1373
                                 6E  D6 0012B          INCL    SCAN_PTR                                    ; 1374
                              5A     02  C0 0012D      ADDL2   #2, I                                       ; 1375
                                 53  11 00130          BRB     22$                                         ; 1367
                  00000041    8F     50  D1 00132 16$:  CMPL    CH, #65                                     ; 1382
                                 09  19 00139          BLSS    17$
                  0000005A    8F     50  D1 0013B      CMPL    CH, #90
                                 45  15 00142          BLEQ    23$
                  00000061    8F     50  D1 00144 17$:  CMPL    CH, #97
                                 09  19 0014B          BLSS    19$
                  0000007A    8F     50  D1 0014D      CMPL    CH, #122
                                 33  15 00154 18$:     BLEQ    23$
                              59 04  A6  9E 00156 19$:  MOVAB   4(R6), R9                                   ; 1390
                              58 00000000'   EF 9E 0015A  MOVAB   SORT_STR, R8
59                20 00000000'   EF     04  2C 00161      MOVC5   #4, P.AAA, #32, R9, (R8)
                                 68          0016A
                                 0C  18 0016B          BGEQ    20$
                              58     04  C0 0016D      ADDL2   #4, R8
```

```
                59              04  C2 00170          SUBL2   #4, R9
     59    20   67              56  2C 00173          MOVC5   LEN, (PTR), #32, R9, (R8)
                                 68     00178
                56              04  C0 00179 20$:     ADDL2   #4, LEN                          ;1392
                57 00000000'    EF  9E 0017C          MOVAB   SORT_STR, PTR                    ;1393
                                04  11 00183 21$:     BRB     23$                              ;1378
          91    5A              5B  F3 00185 22$:     AOBLEQ  R11, I, 15$                      ;1360
             08 AC              56  D1 00189 23$:     CMPL    LEN, I_LEN                       ;1411
                                0E  13 0018D          BEQL    24$
    00000000' EF                56  D0 0018F          MOVL    LEN, SORT_LEN                    ;1418
    00000000' EF                57  D0 00196          MOVL    PTR, SORT_PTR                    ;1419
                01 00000000G    EF  E9 0019D 24$:     BLBC    CMDBLK+1, 25$                    ;1422
                                04  001A4           RET
                                51  7C 001A5 25$:     CLRQ    RINTES_LEN                       ;1434
                54              57  D0 001A7          MOVL    PTR, SCAN_PTR                    ;1435
    00000000' EF 00000000'      EF  9E 001AA          MOVAB   SORT_STR, SORT_PTR               ;1437
                 00000000'      EF  D4 001B5          CLRL    SORT_LEN                         ;1438
                                53  D4 001BB          CLRL    I                                ;1440
                                4F  11 001BD          BRB     28$
                50              84  9A 001BF 26$:     MOVZBL  (SCAN_PTR)+, CH                  ;1445
    00000000G   8F              50  D1 001C2          CMPL    CH, #RINTES                      ;1447
                                45  12 001C9          BNEQ    29$
                                51  D5 001CB          TSTL    RINTES_LEN                       ;1453
                                0E  12 001CD          BNEQ    27$
                51 00000000'    EF  D0 001CF          MOVL    SORT_LEN, RINTES_LEN             ;1460
                52 00000000'    EF  D0 001D6          MOVL    SORT_PTR, RINTES_PTR             ;1461
    00000000'   FF              50  90 001DD 27$:     MOVB    CH, @SORT_PTR                    ;1464
                 00000000'      EF  D6 001E4          INCL    SORT_PTR
    00000000'   FF              84  90 001EA          MOVB    (SCAN_PTR)+, @SORT_PTR           ;1465
                 00000000'      EF  D6 001F1          INCL    SORT_PTR
    00000000'   FF              84  90 001F7          MOVB    (SCAN_PTR)+, @SORT_PTR           ;1466
                 00000000'      EF  D6 001FE          INCL    SORT_PTR
    00000000'   EF              03  C0 00204          ADDL2   #3, SORT_LEN                     ;1467
                53              02  C0 0020B          ADDL2   #2, I                            ;1468
                                3C  11 0020E 28$:     BRB     33$                              ;1447
                20              50  D1 00210 29$:     CMPL    CH, #32                          ;1472
                                0E  13 00213          BEQL    30$
    00000000G   8F              50  D1 00215          CMPL    CH, #TAB
                                05  13 0021C          BEQL    30$
                2D              50  D1 0021E          CMPL    CH, #45
                                14  12 00221          BNEQ    31$
                                52  D5 00223 30$:     TSTL    RINTES_PTR                       ;1478
                                25  13 00225          BEQL    33$
    00000000'   EF              52  D0 00227          MOVL    RINTES_PTR, SORT_PTR             ;1485
    00000000'   EF              51  D0 0022E          MOVL    RINTES_LEN, SORT_LEN             ;1486
                                13  11 00235          BRB     32$                              ;1488
    00000000'   FF              50  90 00237 31$:     MOVB    CH, @SORT_PTR                    ;1497
                 00000000'      EF  D6 0023E          INCL    SORT_PTR
                 00000000'      EF  D6 00244          INCL    SORT_LEN                         ;1498
                                51  7C 0024A 32$:     CLRQ    RINTES_LEN                       ;1501
   FF6D      53                 56  F1 0024C 33$:     ACBL    LEN, #T, I, 26$                  ;1440
    00000000'   EF 00000000'    EF  9E 00252          MOVAB   SORT_STR, SORT_PTR               ;1506
                                04  0025D           RET                                      ;1509

; Routine Size:  606 bytes,    Routine Base:  $CODE$ + 011D
```

```
;  709          1510  1 %SBTTL 'FIND_POS -- Locate position for insertion'
;  710          1511  1
;  711          1512  1 ROUTINE FIND_POS (STG_PTR, STG_LEN, SUB_CNT, LAST, XTN) : NOVALUE =
;  712          1513  1
;  713          1514  1 !++
;  714          1515  1 ! FUNCTIONAL DESCRIPTION:
;  715          1516  1 !
;  716          1517  1 !       Locate the proper position in the master list for placing a new
;  717          1518  1 !       item.  Also make sure the item is not a complete duplicate of an
;  718          1519  1 !       existing entry.
;  719          1520  1 !
;  720          1521  1 ! FORMAL PARAMETERS:
;  721          1522  1 !
;  722          1523  1 !       STG_PTR - Address of input text.
;  723          1524  1 !       STG_LEN - Length of input text.
;  724          1525  1 !       SUB_CNT - Sub-index level (0 to n)
;  725          1526  1 !       LAST    - TRUE if this is the last call to FIND_POS for this entry
;  726          1527  1 !       XTN     - Transaction number if LAST = TRUE
;  727          1528  1 !
;  728          1529  1 ! IMPLICIT INPUTS:
;  729          1530  1 !
;  730          1531  1 !       CELL            - Characteristics of current position in list
;  731          1532  1 !       SORT_LEN        - Length of sort string if any
;  732          1533  1 !       SORT_PTR        - Pointer to sort string if any
;  733          1534  1 !
;  734          1535  1 ! IMPLICIT OUTPUTS:
;  735          1536  1 !
;  736          1537  1 !       CELL            - set up for insertion
;  737          1538  1 !
;  738          1539  1 ! ROUTINE VALUE:
;  739          1540  1 ! COMPLETION CODES:
;  740          1541  1 !
;  741          1542  1 !       NONE.
;  742          1543  1 !
;  743          1544  1 ! SIDE EFFECTS:
;  744          1545  1 !
;  745          1546  1 !       NONE
;  746          1547  1 !
;  747          1548  1 !--
;  748          1549  1
;  749          1550  2     BEGIN
;  750          1551  2
;  751          1552  2     LOCAL
;  752          1553  2         LINE_PTR,
;  753          1554  2         LINE_LEN;
;  754          1555  2
;  755          1556  2     IF .SORT_LEN NEQ 0
;  756          1557  2     THEN
;  757          1558  3         BEGIN
;  758          1559  3         !
;  759          1560  3         ! Have a sort string to use
;  760          1561  3         !
;  761          1562  3         LINE_PTR = .SORT_PTR;
;  762          1563  3         LINE_LEN = .SORT_LEN;
;  763          1564  3         END
;  764          1565  2     ELSE
;  765          1566  3         BEGIN
```

```
;  766     1567  3          !
;  767     1568  3          ! no sort string - use entry text
;  768     1569  3          !
;  769     1570  3          LINE_PTR = .STG_PTR;
;  770     1571  3          LINE_LEN = .STG_LEN;
;  771     1572  3          END;
;  772     1573  3
;  773     1574  3      !
;  774     1575  3      ! Skip the bucket positioning for sub-indexes
;  775     1576  3      !
;  776     1577  3      IF .SUB_CNT EQL 0
;  777     1578  3      THEN
;  778     1579  3          BEGIN
;  779     1580  3          !
;  780     1581  3          ! The first character that is not a special sequence determines
;  781     1582  3          ! the bucket number.
;  782     1583  3          !
;  783     1584  3          LOCAL
;  784     1585  3              BUCKET_NUMBER,
;  785     1586  3              SUB_BUCKET;
;  786     1587  3
;  787     1588  3          BUCKET_NUMBER = FIND_BUCKET (LINE_LEN, LINE_PTR);
;  788     1589  3
;  789     1590  3          IF .BUCKET_NUMBER NEQ 0
;  790     1591  3          THEN
;  791     1592  3              !
;  792     1593  3              ! Use the second character in the string to determine the
;  793     1594  3              ! sub-bucket number unless the first character in the string was a
;  794     1595  3              ! nonalphabetic.
;  795     1596  3              !
;  796     1597  3              SUB_BUCKET = FIND_BUCKET (LINE_LEN, LINE_PTR)
;  797     1598  3          ELSE
;  798     1599  3              !
;  799     1600  3              ! Nonalphabetic characters are always sorted using a single bucket
;  800     1601  3              ! because the 'squared bucket' algorithm does not work for them.
;  801     1602  3              !
;  802     1603  3              SUB_BUCKET = 0;
;  803     1604  3
;  804     1605  3          !
;  805     1606  3          ! Now remember all of the information needed for future use.
;  806     1607  3          !
;  807     1608  3          CELL [C$A_HEAD] = BUCKET [.BUCKET_NUMBER, .SUB_BUCKET];
;  808     1609  3          CELL [C$A_CURR] = .BUCKET [.BUCKET_NUMBER, .SUB_BUCKET];
;  809     1610  3          CELL [C$A_PREV] = 0;
;  810     1611  3          CELL [C$V_IDNS] = FALSE;
;  811     1612  3          END;
;  812     1613  3
;  813     1614  3      !
;  814     1615  3      ! Now find the proper position for insertion
;  815     1616  3      !
;  816     1617  3      REPEAT
;  817     1618  3          BEGIN
;  818     1619  3
;  819     1620  3          LOCAL
;  820     1621  3              CUR_CELL : REF $XE_BLOCK;
;  821     1622  3
;  822     1623  3          !
```

```
;  823          1624  3          ! Point to data in storage
;  824          1625  3          !
;  825          1626  3          CUR_CELL = .CELL [C$A_CURR];
;  826          1627  3          !
;  827          1628  3          ! If this is the last item, return current position
;  828          1629  3          !
;  829          1630  3          IF (.CUR_CELL[XE$A_NEXT] EQL 0) AND (.SUB_CNT EQL .CUR_CELL [XE$H_SUBC]) THEN RETURN;
;  830          1631  3          !
;  831          1632  3          ! See if we are at the correct position for an insertion
;  832          1633  3          !
;  833          1634  3          IF .SUB_CNT GTR .CUR_CELL [XE$H_SUBC] THEN RETURN;
;  834          1635  3          IF ENTRY_CMP (.STG_PTR, .STG_LEN, .LAST, .XTN, .SUB_CNT) THEN RETURN;
;  835          1636  3          !
;  836          1637  3          ! Make sure we still point at original data
;  837          1638  3          !
;  838          1639  3          CUR_CELL = .CELL [C$A_CURR];
;  839          1640  3          !
;  840          1641  3          ! Advance to next location
;  841          1642  3          !
;  842          1643  3          CELL [C$A_CURR] = .CUR_CELL [XE$A_NEXT];
;  843          1644  3          END;
;  844          1645  2
;  845          1646  1      END;                        !End of FIND_POS


                                007C 00000 FIND_POS:
                                        .WORD   Save R2,R3,R4,R5,R6                     ; 1512
                    56 00000000G    EF  9E 00002        MOVAB   BUCKET, R6              ;
                    55 00000000V    EF  9E 00009        MOVAB   FIND_BUCKET, R5         ;
                    54 00000000'    EF  9E 00010        MOVAB   CELL, R4                ;
                    5E              08  C2 00017        SUBL2   #8, SP                  ;
                    50      04C4    C4  D0 0001A        MOVL    SORT_LEN, R0            ; 1556
                                    0B  13 0001F        BEQL    1$                      ;
                    6E      04C0    C4  D0 00021        MOVL    SORT_PTR, LINE_PTR      ; 1562
                04  AE              50  D0 00026        MOVL    R0, LINE_LEN            ; 1563
                                    04  11 0002A        BRB     2$                      ; 1556
                    6E          04  AC  7D 0002C 1$:    MOVQ    STG_PTR, LINE_PTR       ; 1570
                    53          0C  AC  D0 00030 2$:    MOVL    SUB_CNT, R3             ; 1577
                                    33  12 00034        BNEQ    5$                      ;
                                    5E  DD 00036        PUSHL   SP                      ; 1588
                                08  AE  9F 00038        PUSHAB  LINE_LEN                ;
                                    02  FB 0003B        CALLS   #2, FIND_BUCKET         ;
                    52              50  D0 0003E        MOVL    R0, BUCKET_NUMBER       ;
                                    0D  13 00041        BEQL    3$                      ; 1590
                                    5E  DD 00043        PUSHL   SP                      ; 1597
                                08  AE  9F 00045        PUSHAB  LINE_LEN                ;
                                    02  FB 00048        CALLS   #2, FIND_BUCKET         ;
                    51              50  D0 0004B        MOVL    R0, SUB_BUCKET          ;
                                    02  11 0004E        BRB     4$                      ;
```

```
                                                 51  D4 00050 3$:    CLRL    SUB_BUCKET                                  ; 1603
                    50          52               1B  C5 00052 4$:    MULL3   #27, BUCKET_NUMBER, R0                      ; 1608
                                50               51  C0 00056        ADDL2   SUB_BUCKET, R0                              ;
                          08    A4           6640  DE 00059        MOVAL   BUCKET[R0], CELL+8                          ;
                                64           6640  D0 0005E        MOVL    BUCKET[R0], CELL                            ; 1609
                             04 A4              D4 00062        CLRL    CELL+4                                      ; 1610
                          0C    A4               01  8A 00065        BICB2   #1, CELL+12                                 ; 1611
                                52               64  D0 00069 5$:    MOVL    CELL, CUR_CELL                              ; 1626
                             04 A2              D5 0006C        TSTL    4(CUR_CELL)                                 ; 1631
                                                 08  12 0006F        BNEQ    6$                                          ;
        53      18  A2                 10       00  EC 00071        CMPV    #0, #16, 24(CUR_CELL), R3                   ;
                                                 25  13 00077        BEQL    7$                                          ;
        53      18  A2                 10       00  EC 00079 6$:    CMPV    #0, #16, 24(CUR_CELL), R3                   ; 1636
                                                 1D  19 0007F        BLSS    7$                                          ;
                                                 53  DD 00081        PUSHL   R3                                          ; 1638
                                7E       10      AC  7D 00083        MOVQ    LAST, -(SP)                                 ;
                                7E       04      AC  7D 00087        MOVQ    STG_PTR, -(SP)                              ;
                    00000000V   EF               05  FB 0008B        CALLS   #5, ENTRY_CMP                               ;
                                09               50  E8 00092        BLBS    R0, 7$                                      ;
                                52               64  D0 00095        MOVL    CELL, CUR_CELL                              ; 1643
                                64           04  A2  D0 00098        MOVL    4(CUR_CELL), CELL                           ; 1648
                                                 CB  11 0009C        BRB     5$                                          ; 1612
                                                     04 0009E 7$:    RET                                                 ; 1651

; Routine Size:  159 bytes,    Routine Base:  $CODE$ + 037B
```

```
;   852          1652  1 %SBTTL 'FIND_BUCKET -- Get bucket number'
;   853          1653  1 ROUTINE FIND_BUCKET (LEN_A, PTR_A) =
;   854          1654  1 !++
;   855          1655  1 !
;   856          1656  1 ! FUNCTIONAL DESCRIPTION:
;   857          1657  1 !
;   858          1658  1 !       This routine is called to determine the bucket number of the first
;   859          1659  1 !       character in a string which is not a special sequence
;   860          1660  1 !
;   861          1661  1 ! FORMAL PARAMETERS:
;   862          1662  1 !
;   863          1663  1 !       LEN_A   - Address of a variable which contains the length of the string
;   864          1664  1 !                 The value is updated to reflect the number of unscanned
;   865          1665  1 !                 characters in the string.
;   866          1666  1 !       PTR_A   - Address of a variable which contains a CH$PTR to the string
;   867          1667  1 !                 The value is updated to point to the first unscanned
;   868          1668  1 !                 character in the string.
;   869          1669  1 !
;   870          1670  1 ! IMPLICIT INPUTS:
;   871          1671  1 !
;   872          1672  1 !       None
;   873          1673  1 !
;   874          1674  1 ! IMPLICIT OUTPUTS:
;   875          1675  1 !
;   876          1676  1 !       None
;   877          1677  1 !
;   878          1678  1 ! ROUTINE VALUE:
;   879          1679  1 ! COMPLETION CODES:
;   880          1680  1 !
;   881          1681  1 !       Returns a value from 0 to 26 indicating the bucket number.
;   882          1682  1 !       (0 = nonalpha, 1 = A, ... 26 = Z)
;   883          1683  1 !
;   884          1684  1 ! SIDE EFFECTS:
;   885          1685  1 !
;   886          1686  1 !       None
;   887          1687  1 !--
;   888          1688  2     BEGIN
;   889          1689  2
;   890          1690  2     LOCAL
;   891          1691  2         CH;
;   892          1692  2
;   893          1693  2     BIND
;   894          1694  2         LEN = .LEN_A,
;   895          1695  2         PTR = .PTR_A;
;   896          1696  2
;   897          1697  2     CH = 0;
;   898          1698  2
;   899          1699  2     WHILE .LEN GTR 0 DO
;   900          1700  2         BEGIN
;   901          1701  2         !
;   902          1702  2         ! Get the first character that is not a special sequence
;   903          1703  2         !
;   904          1704  2         CH = CH$RCHAR_A (PTR);
;   905          1705  2         LEN = .LEN - 1;
;   906          1706  2
;   907          1707  3         IF .CH EQL RINTES
;   908          1708  3         THEN
```

```
;  909          1709  4              BEGIN
;  910          1710  4              !
;  911          1711  4              ! Skip special sequence
;  912          1712  4              !
;  913          1713  4              CH$RCHAR_A (PTR);
;  914          1714  4              CH$RCHAR_A (PTR);
;  915          1715  4              LEN = .LEN - 2;
;  916          1716  4              END
;  917          1717  3          ELSE
;  918          1718  4              BEGIN
;  919          1719  4              !
;  920          1720  4              ! Some other character
;  921          1721  4              !
;  922          1722  4              IF LOWER_LETTER (.CH) THEN CH = UPPER_CASE (.CH);
;  923          1723  4
;  924          1724  4              EXITLOOP;
;  925          1725  3              END;
;  926          1726  3
;  927          1727  2          END;
;  928          1728  2
;  929          1729  2      !
;  930          1730  2      ! Using the first non-special character, figure out which index
;  931          1731  2      ! bucket is the right one to look into. Buckets 1 through 26 are
;  932          1732  2      ! alphabetic, and all other characters belong in bucket 0.
;  933          1733  2      !
;  934          1734  2      RETURN (IF (.CH GEQ %C'A') AND (.CH LEQ %C'Z') THEN (.CH - %C'A' + 1) ELSE 0);
;  935          1735  1      END;


                                0004 00000 FIND_BUCKET:
                                           .WORD   Save R2                                  ;  1653
                    51    04    AC  D0 00002       MOVL    LEN_A, R1                       ;  1694
                                50  D4 00006       CLRL    CH                              ;  1697
                                61  D5 00008 1$:    TSTL    (R1)                            ;  1699
                                35  15 0000A       BLEQ    3$                              ;
                    52    08    BC  D0 0000C       MOVL    @PTR_A, R2                      ;  1704
                    50          62  9A 00010       MOVZBL  (R2), CH                        ;
                          08    BC  D6 00013       INCL    @PTR_A                          ;
                                61  D7 00016       DECL    (R1)                            ;  1705
          00000000G 8F          50  D1 00018       CMPL    CH, #RINTES                     ;  1707
                                0B  12 0001F       BNEQ    2$                              ;
                          08    BC  D6 00021       INCL    @PTR_A                          ;  1713
                          08    BC  D6 00024       INCL    @PTR_A                          ;  1714
                    61          02  C2 00027       SUBL2   #2, (R1)                        ;  1715
                                DC  11 0002A       BRB     1$                              ;  1707
          00000061  8F          50  D1 0002C 2$:    CMPL    CH, #97                         ;  1722
                                0C  19 00033       BLSS    3$                              ;
          0000007A  8F          50  D1 00035       CMPL    CH, #122                        ;
                                03  14 0003C       BGTR    3$                              ;
                    50          20  C2 0003E       SUBL2   #32, CH                         ;
          00000041  8F          50  D1 00041 3$:    CMPL    CH, #65                         ;  1734
                                0E  19 00048       BLSS    4$                              ;
          0000005A  8F          50  D1 0004A       CMPL    CH, #90                         ;
                                05  14 00051       BGTR    4$                              ;
```

```
                                50       C0   A0  9E 00053          MOVAB   -64(R0), R0                     ;
                                                 04 00057          RET                                     ;
                                          50     04 00058 48:      CLRL    R0                              ;
                                                 04 0005A          RET                                     ; 1735

; Routine Size:  91 bytes,    Routine Base:  $CODE$ + 041A
```

```
;   937         1736  1 %SBTTL 'INSERT_INX -- Insert index item into list'
;   938         1737  1
;   939         1738  1 ROUTINE INSERT_INX (STRING, LNGTH, SUB_CNT, XTN, BAR, ENT_LEN, ENT_PTR) : NOVALUE =
;   940         1739  1
;   941         1740  1 !++
;   942         1741  1 ! FUNCTIONAL DESCRIPTION:
;   943         1742  1 !
;   944         1743  1 !       Insert an index item into the index list.
;   945         1744  1 !
;   946         1745  1 ! FORMAL PARAMETERS:
;   947         1746  1 !
;   948         1747  1 !       STRING  - CH$PTR to the string associated with the item.
;   949         1748  1 !                 (zero is allowed).
;   950         1749  1 !       LNGTH   - Length of the passed string.
;   951         1750  1 !       SUB_CNT - Sub-index level of item (0 to n)
;   952         1751  1 !       XTN     - Transaction number of the page associated with this
;   953         1752  1 !                 index item.
;   954         1753  1 !       BAR     - Change bar flag
;   955         1754  1 !       ENT_LEN - Length of whole index entry
;   956         1755  1 !       ENT_PTR - CH$PTR to whole index entry string
;   957         1756  1 !
;   958         1757  1 ! IMPLICIT INPUTS:
;   959         1758  1 !
;   960         1759  1 !       CELL    - Information table about current position in list.
;   961         1760  1 !       BOOKID  - Master index book ident string address
;   962         1761  1 !       SORT_LEN- Length of sort string if any
;   963         1762  1 !       SORT_PTR- Pointer to sort string if any
;   964         1763  1 !
;   965         1764  1 ! IMPLICIT OUTPUTS:
;   966         1765  1 !
;   967         1766  1 !       NONE
;   968         1767  1 !
;   969         1768  1 ! ROUTINE VALUE:
;   970         1769  1 ! COMPLETION CODES:
;   971         1770  1 !
;   972         1771  1 !       NONE
;   973         1772  1 !
;   974         1773  1 ! SIDE EFFECTS:
;   975         1774  1 !
;   976         1775  1 !       NONE
;   977         1776  1 !
;   978         1777  1 !--
;   979         1778  1
;   980         1779  2     BEGIN
;   981         1780  2     LOCAL
;   982         1781  2         XMREF : $XM_BLOCK;
;   983         1782  2
;   984         1783  2     !
;   985         1784  2     ! Build book reference entry
;   986         1785  2     !
;   987         1786  2     XMREF [XM$A_LINK] = 0;
;   988         1787  2     XMREF [XM$A_BOOK] = .BOOKID;
;   989         1788  2
;   990         1789  2     !
;   991         1790  2     ! Check for existing entry
;   992         1791  2     !
;   993         1792  2     IF .CELL [C$V_IDNS]
```

```
;  994     1793  2      THEN
;  995     1794  3          BEGIN
;  996     1795  3          !
;  997     1796  3          ! Identical string
;  998     1797  3          !
;  999     1798  3          LOCAL
; 1000     1799  3              XE_TEMP : REF $XE_BLOCK,
; 1001     1800  3              XM_TEMP : REF $XM_BLOCK;
; 1002     1801  3
; 1003     1802  3          !
; 1004     1803  3          ! Get current cell
; 1005     1804  3          !
; 1006     1805  3          XE_TEMP = .CELL [C$A_CURR];
; 1007     1806  3
; 1008     1807  3          !
; 1009     1808  3          ! Get first entry in book list chain
; 1010     1809  3          !
; 1011     1810  3          XM_TEMP = .XE_TEMP [XE$A_BOOK_LIST];
; 1012     1811  3
; 1013     1812  3          REPEAT
; 1014     1813  4              BEGIN
; 1015     1814  4              !
; 1016     1815  4              ! Walk book list chain until we either find a reference to
; 1017     1816  4              ! the current book or until the end of chain
; 1018     1817  4              !
; 1019     1818  4              IF .XM_TEMP [XM$A_BOOK] EQL .BOOKID THEN EXITLOOP;
; 1020     1819  4
; 1021     1820  4              IF .XM_TEMP [XM$A_LINK] EQL 0
; 1022     1821  4              THEN
; 1023     1822  5                  BEGIN
; 1024     1823  5                  !
; 1025     1824  5                  ! End of chain - insert a new book reference
; 1026     1825  5                  !
; 1027     1826  5                  XM_TEMP [XM$A_LINK] = SAVDAT (XMREF, DS_XM_ENTRY, XM$K_LENGTH);
; 1028     1827  5                  EXITLOOP;
; 1029     1828  5                  END
; 1030     1829  4              ELSE
; 1031     1830  4                  XM_TEMP = .XM_TEMP [XM$A_LINK];
; 1032     1831  3              END;
; 1033     1832  3
; 1034     1833  3          IF .XTN NEQ 0
; 1035     1834  3          THEN
; 1036     1835  4              BEGIN
; 1037     1836  4              !
; 1038     1837  4              ! There is a page pointer, so attach it.
; 1039     1838  4              !
; 1040     1839  4              LOCAL
; 1041     1840  4                  XX_TEMP : REF $XX_BLOCK;
; 1042     1841  4
; 1043     1842  4              IF .XE_TEMP [XE$A_REF] NEQ 0
; 1044     1843  4              THEN
; 1045     1844  5                  BEGIN
; 1046     1845  5                  !
; 1047     1846  5                  ! Entry has references
; 1048     1847  5                  !
; 1049     1848  5                  LOCAL
; 1050     1849  5                      RANGE_BOOK,
```

```
;  1051          1850  5                        RANGE_ACTIVE;
;  1052          1851  5
;  1053          1852  5                    RANGE_ACTIVE = FALSE;           ! Have not seen a BEGIN yet
;  1054          1853  5                    RANGE_BOOK = 0;
;  1055          1854  5                    XX_TEMP = .XE_TEMP [XE$A_REF];  ! Get the start of the chain
;  1056          1855  5
;  1057          1856  5                    REPEAT
;  1058          1857  6                        BEGIN
;  1059          1858  6                        !
;  1060          1859  6                        ! Find the chain end
;  1061          1860  6                        !
;  1062          1861  6                        XX_TEMP = .XX_TEMP;
;  1063          1862  6
;  1064          1863  6                        IF .XX_TEMP [XX$V_BEGIN]
;  1065          1864  6                        THEN
;  1066          1865  7                            BEGIN
;  1067          1866  7                            !
;  1068          1867  7                            ! Beginning of page range
;  1069          1868  7                            !
;  1070          1869  7                            RANGE_ACTIVE = TRUE;
;  1071          1870  7                            RANGE_BOOK = .XX_TEMP [XX$A_BOOK];      ! Save book identifier.
;  1072          1871  6                            END;
;  1073          1872  6
;  1074          1873  6                        IF .XX_TEMP [XX$V_END]
;  1075          1874  6                        OR .XX_TEMP [XX$A_BOOK] NEQ .RANGE_BOOK
;  1076          1875  6                        THEN
;  1077          1876  6                            !
;  1078          1877  6                            ! A range ends when either an END is encountered
;  1079          1878  6                            ! or when we switch books
;  1080          1879  6                            !
;  1081          1880  6                            RANGE_ACTIVE = FALSE;
;  1082          1881  6
;  1083          1882  6                        !
;  1084          1883  6                        ! Check for end of chain
;  1085          1884  6                        !
;  1086          1885  6                        IF  .XX_TEMP [XX$A_LINK] NEQ 0
;  1087          1886  6                        THEN
;  1088          1887  6                            XX_TEMP = .XX_TEMP [XX$A_LINK]
;  1089          1888  6                        ELSE
;  1090          1889  6                            EXITLOOP;
;  1091          1890  6
;  1092          1891  5                        END;
;  1093          1892  5
;  1094          1893  5                    IF .RANGE_ACTIVE
;  1095          1894  5                    THEN
;  1096          1895  6                        BEGIN
;  1097          1896  6                        !
;  1098          1897  6                        ! Saw a BEGIN with no END
;  1099          1898  6                        !
;  1100          1899  6                        IF .XPLBLK [XPL$V_VALID] AND .XPLBLK [XPL$V_BEGIN]
;  1101          1900  6                        THEN
;  1102          1901  7                            BEGIN
;  1103          1902  7                            !
;  1104          1903  7                            ! Have a BEGIN inside a BEGIN
;  1105          1904  7                            !
;  1106        L 1905  7 %IF %BLISS (BLISS32)
;  1107          1906  7 %THEN                                               ! Signal errors for BLISS32
```

```
;  1108          1907  7
;  1109          1908  7                          SIGNAL (INDEX$_DUPBEGIN);
;  1110          1909  7
;  1111        U 1910  7 %ELSE                                                 ! Use $XPO_PUT_MSG otherwise
;  1112        U 1911  7
;  1113        U 1912  7                          $XPO_PUT_MSG (SEVERITY = WARNING,
;  1114        U 1913  7                              STRING = 'duplicate .XPLUS (BEGIN) -- inserted as .XPLUS ()');
;  1115        U 1914  7
;  1116          1915  7 %FI
;  1117          1916  7
;  1118          1917  7                          DMPENT (.ENT_LEN, .ENT_PTR);
;  1119          1918  7                          XPLBLK [XPL$V_BEGIN] = FALSE;
;  1120          1919  6                          END;
;  1121          1920  6                      END
;  1122          1921  5                  ELSE
;  1123          1922  5                      !
;  1124          1923  5                      ! Have no unmatched BEGIN's
;  1125          1924  5                      !
;  1126          1925  5                      IF .XPLBLK [XPL$V_VALID] AND .XPLBLK [XPL$V_END]
;  1127          1926  5                      THEN
;  1128          1927  6                          BEGIN
;  1129          1928  6                          !
;  1130          1929  6                          ! Have an END with no BEGIN
;  1131          1930  6                          !
;  1132        L 1931  6 %IF %BLISS (BLISS32)
;  1133          1932  6 %THEN                                                 ! Signal errors for BLISS32
;  1134          1933  6
;  1135          1934  6                          SIGNAL (INDEX$_NOBEGIN);
;  1136          1935  6
;  1137        U 1936  6 %ELSE                                                 ! Use $XPO_PUT_MSG otherwise
;  1138        U 1937  6
;  1139        U 1938  6                          $XPO_PUT_MSG (SEVERITY = WARNING,
;  1140        U 1939  6                              STRING = '.XPLUS (END) with no .XPLUS (BEGIN) -- inserted as .XPLUS ()');
;  1141        U 1940  6
;  1142          1941  6 %FI
;  1143          1942  6
;  1144          1943  6                          DMPENT (.ENT_LEN, .ENT_PTR);
;  1145          1944  6                          XPLBLK [XPL$V_END] = FALSE;
;  1146          1945  5                          END;
;  1147          1946  5
;  1148          1947  5                      !
;  1149          1948  5                      ! Add new pointer to entry and update it in memory
;  1150          1949  5                      !
;  1151          1950  5                      XX_TEMP [XX$A_LINK] = INSERT_REF (.XTN);
;  1152          1951  5                      END
;  1153          1952  4                  ELSE
;  1154          1953  5                      BEGIN
;  1155          1954  5                      !
;  1156          1955  5                      ! Entry has no references.
;  1157          1956  5                      !
;  1158          1957  5                      IF .XPLBLK [XPL$V_VALID] AND .XPLBLK [XPL$V_END]
;  1159          1958  5                      THEN
;  1160          1959  6                          BEGIN
;  1161          1960  6                          !
;  1162          1961  6                          ! Have an END with no BEGIN
;  1163          1962  6                          !
;  1164        L 1963  6 %IF %BLISS (BLISS32)
```

```
;  1165          1964  6 %THEN                                                 ! Signal errors for BLISS32
;  1166          1965  6
;  1167          1966  6                     SIGNAL (INDEX$_NOBEGIN);
;  1168          1967  6
;  1169        U 1968  6 %ELSE                                                 ! Use $XPO_PUT_MSG otherwise
;  1170        U 1969  6
;  1171        U 1970  6                     $XPO_PUT_MSG (SEVERITY = WARNING,
;  1172        U 1971  6                         STRING = '.XPLUS (END) with no .XPLUS (BEGIN) -- inserted as .XPLUS ()');
;  1173        U 1972  6
;  1174          1973  6 %FI
;  1175          1974  6
;  1176          1975  6                     DMPENT (.ENT_LEN, .ENT_PTR);
;  1177          1976  6                     XPLBLK [XPL$V_END] = FALSE;
;  1178          1977  5                     END;
;  1179          1978  5
;  1180          1979  5                 !
;  1181          1980  5                 ! Point entry to reference and update it in memory.
;  1182          1981  5                 !
;  1183          1982  5                 XE_TEMP [XE$A_REF] = INSERT_REF (.XTN);
;  1184          1983  4                 END;
;  1185          1984  4
;  1186          1985  3             END;
;  1187          1986  3         END
;  1188          1987  2     ELSE
;  1189          1988  3         BEGIN
;  1190          1989  3         !
;  1191          1990  3         ! String is different, insert new string
;  1192          1991  3         !
;  1193          1992  3         LOCAL
;  1194          1993  3             REF_PTR,
;  1195          1994  3             LAST_CELL,
;  1196          1995  3             NEXT_CELL,
;  1197          1996  3             TEMP      : REF $XE_BLOCK,
;  1198          1997  3             TEMP1,
;  1199          1998  3             TEMP_CELL : $XE_BLOCK;
;  1200          1999  3
;  1201          2000  3         !
;  1202          2001  3         ! Get links to chain
;  1203          2002  3         !
;  1204          2003  3         TEMP = .CELL [C$A_CURR];
;  1205          2004  3
;  1206          2005  3         IF .SUB_CNT EQL .TEMP [XE$H_SUBC]
;  1207          2006  3         THEN
;  1208          2007  4             BEGIN
;  1209          2008  4             NEXT_CELL = .CELL [C$A_CURR];
;  1210          2009  4             LAST_CELL = .TEMP [XE$A_PREV]
;  1211          2010  4             END
;  1212          2011  3         ELSE
;  1213          2012  4             BEGIN
;  1214          2013  4             NEXT_CELL = 0;
;  1215          2014  4             LAST_CELL = .CELL [C$A_CURR]
;  1216          2015  4             END;
;  1217          2016  3
;  1218          2017  3         IF .XTN NEQ 0
;  1219          2018  3         THEN
;  1220          2019  4             BEGIN
;  1221          2020  4             !
```

```
;  1222     2021  4              ! Have a page reference
;  1223     2022  4              !
;  1224     2023  4              IF .XPLBLK [XPL$V_VALID] AND .XPLBLK [XPL$V_END]
;  1225     2024  4              THEN
;  1226     2025  5                  BEGIN
;  1227     2026  5                  !
;  1228     2027  5                  ! Have an END with no BEGIN
;  1229     2028  5                  !
;  1230   L 2029  5 %IF %BLISS (BLISS32)
;  1231     2030  5 %THEN                                          ! Signal errors for BLISS32
;  1232     2031  5
;  1233     2032  5                  SIGNAL (INDEX$_NOBEGIN);
;  1234     2033  5
;  1235   U 2034  5 %ELSE                                          ! Use $XPO_PUT_MSG otherwise
;  1236   U  2035  5
;  1237   U  2036  5                  $XPO_PUT_MSG (SEVERITY = WARNING,
;  1238   U  2037  5                      STRING = '.XPLUS (END) with no .XPLUS (BEGIN) -- inserted as .XPLUS ()');
;  1239   U 2038  5
;  1240     2039  5 %FI
;  1241     2040  5
;  1242     2041  5                  DMPENT (.ENT_LEN, .ENT_PTR);
;  1243     2042  5                  XPLBLK [XPL$V_END] = FALSE;
;  1244     2043  4                  END;
;  1245     2044  4
;  1246     2045  4              REF_PTR = INSERT_REF (.XTN);
;  1247     2046  4              END
;  1248     2047  3          ELSE
;  1249     2048  3              REF_PTR = 0;
;  1250     2049  3
;  1251     2050  3          !
;  1252     2051  3          ! Start to set up new entry
;  1253     2052  3          !
;  1254     2053  3          TEMP_CELL [XE$A_PREV] = .LAST_CELL;
;  1255     2054  3          TEMP_CELL [XE$A_NEXT] = .NEXT_CELL;
;  1256     2055  3          TEMP_CELL [XE$H_SUBC] = .SUB_CNT;
;  1257     2056  3          TEMP_CELL [XE$V_BARS] = .BAR;
;  1258     2057  3          TEMP_CELL [XE$A_REF] = .REF_PTR;
;  1259     2058  3          TEMP_CELL [XE$A_SUBX] = 0;
;  1260     2059  3          TEMP_CELL [XE$A_BOOK_LIST] = SAVDAT (XMREF, DS_XM_ENTRY, XM$K_LENGTH);
;  1261     2060  3          !
;  1262     2061  3          ! Remember text string
;  1263     2062  3          !
;  1264     2063  3          IF .STRING NEQ 0
;  1265     2064  3          THEN
;  1266     2065  3              TEMP_CELL [XE$A_TEXT] = SAVDAT (.STRING, DS_X_STRING, .LNGTH)
;  1267     2066  3          ELSE
;  1268     2067  3              TEMP_CELL [XE$A_TEXT] = 0;
;  1269     2068  3          !
;  1270     2069  3          ! Save sort string if there is one
;  1271     2070  3          !
;  1272     2071  3          IF .SORT_LEN NEQ 0
;  1273     2072  3          THEN
;  1274     2073  3              TEMP_CELL [XE$A_SORT_AS] = SAVDAT (.SORT_PTR, DS_X_STRING, .SORT_LEN)
;  1275     2074  3          ELSE
;  1276     2075  3              TEMP_CELL [XE$A_SORT_AS] = 0;
;  1277     2076  3
;  1278     2077  3
```

```
;  1279     2078  3
;  1280     2079  3          !
;  1281     2080  3          ! Now put away the entry proper
;  1282     2081  3          !
;  1283     2082  3          TEMP1 = SAVDAT (TEMP_CELL, DS_X_ENTRY, XE$K_LENGTH);
;  1284     2083  3
;  1285     2084  3          !
;  1286     2085  3          ! Link to previous entry
;  1287     2086  3          !
;  1288     2087  3          IF .LAST_CELL NEQ 0
;  1289     2088  3          THEN
;  1290     2089  4              BEGIN
;  1291     2090  4              TEMP = .LAST_CELL;
;  1292     2091  4
;  1293     2092  4              IF .SUB_CNT NEQ .TEMP [XE$H_SUBC]
;  1294     2093  4              THEN
;  1295     2094  4                  TEMP [XE$A_SUBX] = .TEMP1
;  1296     2095  4              ELSE
;  1297     2096  4                  TEMP [XE$A_NEXT] = .TEMP1;
;  1298     2097  4
;  1299     2098  4              END
;  1300     2099  3          ELSE
;  1301     2100  3              !
;  1302     2101  3              ! Head of List
;  1303     2102  3              !
;  1304     2103  3              .CELL [C$A_HEAD] = .TEMP1;
;  1305     2104  3
;  1306     2105  3          !
;  1307     2106  3          ! Link to the following cell
;  1308     2107  3          !
;  1309     2108  3          IF .NEXT_CELL NEQ 0
;  1310     2109  3          THEN
;  1311     2110  4              BEGIN
;  1312     2111  4              TEMP = .NEXT_CELL;
;  1313     2112  4              TEMP [XE$A_PREV] = .TEMP1;
;  1314     2113  3              END;
;  1315     2114  3
;  1316     2115  3          !
;  1317     2116  3          ! Remember where we left off
;  1318     2117  3          !
;  1319     2118  3          CELL [C$A_CURR] = .TEMP1;
;  1320     2119  3          END
;  1321     2120  3
;  1322     2121  1      END;                        !End of INSERT_INX


                                 0FFC 00000 INSERT_INX:
                                                .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11        ; 1738
                      5B 00000000G   8F   D0 00002          MOVL    #DSRINDEX$_NOBEGIN, R11       ;
                      5A 00000000G   EF   9E 00009          MOVAB   DMPENT, R10                   ;
                      59 00000000G   00   9E 00010          MOVAB   LIB$SIGNAL, R9                ;
                      58 00000000G   EF   9E 00017          MOVAB   SAVDAT, R8                    ;
                      57 00000000'   EF   9E 0001E          MOVAB   CELL, R7                      ;
                      56 00000000G   EF   9E 00025          MOVAB   XPLBLK, R6                    ;
```

```
                                5E              28  C2 0002C           SUBL2   #40, SP                                        ;
                                            20  AE  D4 0002F           CLRL    XMREF                                           ;  1786
                             24 AE 00000000G    EF  D0 00032           MOVL    BOOKID, XMREF+4                                 ;  1787
                                50              67  D0 0003A           MOVL    CELL, R0                                        ;  1805
                                54          10  AC  D0 0003D           MOVL    XTN, R4                                         ;  1833
                                03          0C  A7  E8 00041           BLBS    CELL+12, 1$                                     ;  1792
                                                00C6 31 00045           BRW     15$                                             ;
                                53              50  D0 00048 1$:       MOVL    R0, XE_TEMP                                     ;  1805
                                52          1C  A3  D0 0004B           MOVL    28(XE_TEMP), XM_TEMP                            ;  1810
                         00000000G  EF          04  A2  D1 0004F 2$:    CMPL    4(XM_TEMP), BOOKID                              ;  1818
                                                18  13  00057           BEQL    4$                                              ;
                                                62  D5  00059           TSTL    (XM_TEMP)                                       ;  1820
                                                0F  12  0005B           BNEQ    3$                                              ;
                                                02  DD  0005D           PUSHL   #2                                              ;  1826
                                                02  DD  0005F           PUSHL   #2                                              ;
                                            28  AE  9F  00061           PUSHAB  XMREF                                           ;
                                68              03  FB  00064           CALLS   #3, SAVDAT                                      ;
                                62              50  D0  00067           MOVL    R0, (XM_TEMP)                                   ;
                                                05  11  0006A           BRB     4$                                              ;  1822
                                52              62  D0  0006C 3$:       MOVL    (XM_TEMP), XM_TEMP                              ;  1830
                                                DE  11  0006F           BRB     2$                                              ;  1810
                                                54  D5  00071 4$:       TSTL    R4                                              ;  1833
                                                01  12  00073           BNEQ    5$                                              ;
                                                    04  00075           RET                                                     ;
                 55         66                  00  EF  00076 5$:       EXTZV   #0, #1, XPLBLK, R5                              ;  1899
                                            0C  A3  D5  0007B           TSTL    12(XE_TEMP)                                     ;  1842
                                                6A  13  0007E           BEQL    13$                                             ;
                                                50  7C  00080           CLRQ    RANGE_BOOK                                      ;  1853
                                52          0C  A3  D0  00082           MOVL    12(XE_TEMP), XX_TEMP                            ;  1854
                 07         0A  A2              02  E1  00086 6$:       BBC     #2, 10(XX_TEMP), 7$                             ;  1863
                                51              01  D0  0008B           MOVL    #1, RANGE_ACTIVE                                ;  1869
                                50          0C  A2  D0  0008E           MOVL    12(XX_TEMP), RANGE_BOOK                         ;  1870
                 06         0A  A2              03  E0  00092 7$:       BBS     #3, 10(XX_TEMP), 8$                             ;  1873
                                50          0C  A2  D1  00097           CMPL    12(XX_TEMP), RANGE_BOOK                         ;  1874
                                                02  13  0009B           BEQL    9$                                              ;
                                                51  D4  0009D 8$:       CLRL    RANGE_ACTIVE                                    ;  1880
                                                62  D5  0009F 9$:       TSTL    (XX_TEMP)                                       ;  1885
                                                05  13  000A1           BEQL    10$                                             ;
                                52              62  D0  000A3           MOVL    (XX_TEMP), XX_TEMP                              ;  1887
                                                DE  11  000A6           BRB     6$                                              ;
                                1C              51  E9  000A8 10$:      BLBC    RANGE_ACTIVE, 11$                               ;  1893
                                2F              55  E9  000AB           BLBC    R5, 12$                                         ;  1899
                 2B             66              03  E1  000AE           BBC     #3, XPLBLK, 12$                                 ;
                                    00000000G   8F  DD  000B2           PUSHL   #DSRINDEX$_DUPBEGIN                             ;  1908
                                69              01  FB  000B8           CALLS   #1, LIB$SIGNAL                                  ;
                                7E          18  AC  7D  000BB           MOVQ    ENT_LEN, -(SP)                                  ;  1917
                                6A              02  FB  000BF           CALLS   #2, DMPENT                                      ;
                                66              08  8A  000C2           BICB2   #8, XPLBLK                                      ;  1918
                                                16  11  000C5           BRB     12$                                             ;  1893
                                13              55  E9  000C7 11$:      BLBC    R5, 12$                                         ;  1925
                 0F             66              04  E1  000CA           BBC     #4, XPLBLK, 12$                                 ;
                                                5B  DD  000CE           PUSHL   R11                                             ;  1934
                                69              01  FB  000D0           CALLS   #1, LIB$SIGNAL                                  ;
                                7E          18  AC  7D  000D3           MOVQ    ENT_LEN, -(SP)                                  ;  1943
                                6A              02  FB  000D7           CALLS   #2, DMPENT                                      ;
                                66              10  8A  000DA           BICB2   #16, XPLBLK                                     ;  1944
                                                54  DD  000DD 12$:       PUSHL   R4                                              ;  1950
```

```
                          00000000V  EF                 01  FB 000DF                CALLS   #1, INSERT_REF
                                     62                 50  D0 000E6                MOVL    R0, (XX_TEMP)
                                                        04  000E9                   RET                                 ;
                                     13                 55  E9 000EA 13$:           BLBC    R5, 14$                     ; 1842
                                OF   66                 04  E1 000ED                BBC     #4, XPLBLK, 14$             ; 1957
                                                        5B  DD 000F1                PUSHL   R11                         ;
                                     69                 01  FB 000F3                CALLS   #1, LIB$SIGNAL              ; 1966
                                     7E            18   AC  7D 000F6                MOVQ    ENT_LEN, -(SP)              ;
                                     6A                 02  FB 000FA                CALLS   #2, DMPENT                  ; 1975
                                     66                 10  8A 000FD                BICB2   #16, XPLBLK                 ;
                                                        54  DD 00100 14$:           PUSHL   R4                          ; 1976
                          00000000V  EF                 01  FB 00102                CALLS   #1, INSERT_REF              ; 1982
                                OC   A3                 50  D0 00109                MOVL    R0, 12(XE_TEMP)             ;
                                                        04  0010D                   RET                                 ;
                                     53                 50  D0 0010E 15$:           MOVL    R0, TEMP                    ; 1833
OC   AC        18   A3               10                 00  EC 00111                CMPV    #0, #16, 24(TEMP), SUB_CNT  ; 2003
                                                        08  12 00118                BNEQ    16$                         ; 2005
                                     52                 50  D0 0011A                MOVL    R0, NEXT_CELL               ;
                                     55                 63  D0 0011D                MOVL    (TEMP), LAST_CELL           ; 2008
                                                        05  11 00120                BRB     17$                         ; 2009
                                                        52  D4 00122 16$:           CLRL    NEXT_CELL                   ;
                                     55                 50  D0 00124                MOVL    R0, LAST_CELL               ; 2013
                                                        54  D5 00127 17$:           TSTL    R4                          ; 2014
                                                        21  13 00129                BEQL    19$                         ; 2017
                                     13                 66  E9 0012B                BLBC    XPLBLK, 18$                 ;
                                OF   66                 04  E1 0012E                BBC     #4, XPLBLK, 18$             ; 2023
                                                        5B  DD 00132                PUSHL   R11                         ;
                                     69                 01  FB 00134                CALLS   #1, LIB$SIGNAL              ; 2032
                                     7E            18   AC  7D 00137                MOVQ    ENT_LEN, -(SP)              ;
                                     6A                 02  FB 0013B                CALLS   #2, DMPENT                  ; 2041
                                     66                 10  8A 0013E                BICB2   #16, XPLBLK                 ;
                                                        54  DD 00141 18$:           PUSHL   R4                          ; 2042
                          00000000V  EF                 01  FB 00143                CALLS   #1, INSERT_REF              ; 2045
                                                        02  11 0014A                BRB     20$                         ;
                                                        50  D4 0014C 19$:           CLRL    REF_PTR                     ; 2017
                                     6E                 55  D0 0014E 20$:           MOVL    LAST_CELL, TEMP_CELL        ; 2048
                                04   AE                 52  D0 00151                MOVL    NEXT_CELL, TEMP_CELL+4      ; 2053
                                18   AE            0C   AC  B0 00155                MOVW    SUB_CNT, TEMP_CELL+24       ; 2054
1A   AE        01               00                 14  AC  F0 0015A                INSV    BAR, #0, #1, TEMP_CELL+26   ; 2055
                                0C   AE                 50  D0 00161                MOVL    REF_PTR, TEMP_CELL+12       ; 2056
                                                   08   AE  D4 00165                CLRL    TEMP_CELL+8                 ; 2057
                                                        02  DD 00168                PUSHL   #2                          ; 2058
                                                        02  DD 0016A                PUSHL   #2                          ; 2059
                                                   28   AE  9F 0016C                PUSHAB  XMREF                       ;
                                     68                 03  FB 0016F                CALLS   #3, SAVDAT                  ;
                                1C   AE                 50  D0 00172                MOVL    R0, TEMP_CELL+28            ;
                                                   04   AC  D5 00176                TSTL    STRING                      ; 2064
                                                        11  13 00179                BEQL    21$                         ;
                                                   08   AC  DD 0017B                PUSHL   LNGTH                       ; 2066
                                                        7E  D4 0017E                CLRL    -(SP)                       ;
                                                   04   AC  DD 00180                PUSHL   STRING                      ;
                                     68                 03  FB 00183                CALLS   #3, SAVDAT                  ;
                                10   AE                 50  D0 00186                MOVL    R0, TEMP_CELL+16            ;
                                                        03  11 0018A                BRB     22$                         ;
                                                   10   AE  D4 0018C 21$:           CLRL    TEMP_CELL+16                ; 2068
                                     50            04C4 C7  D0 0018F 22$:           MOVL    SORT_LEN, R0                ; 2073
                                                        11  13 00194                BEQL    23$                         ;
```

```
                                                 50  DD 00196          PUSHL   R0                                      ; 2075
                                                 7E  D4 00198          CLRL    -(SP)                                   ;
                                          04C0   C7  DD 0019A          PUSHL   SORT_PTR                                ;
                                             68  03  FB 0019E          CALLS   #3, SAVDAT                              ;
                                          14 AE  50  D0 001A1          MOVL    R0, TEMP_CELL+20                        ;
                                                 03  11 001A5          BRB     24$                                     ;
                                       14 AE     D4  001A7 23$:       CLRL    TEMP_CELL+20                            ; 2077
                                                 08  DD 001AA 24$:     PUSHL   #8                                      ; 2082
                                                 08  DD 001AC          PUSHL   #8                                      ;
                                       08 AE     9F  001AE          PUSHAB  TEMP_CELL                               ;
                                             68  03  FB 001B1          CALLS   #3, SAVDAT                              ;
                                                 55  D5 001B4          TSTL    LAST_CELL                               ; 2087
                                                 18  13 001B6          BEQL    26$                                     ;
                                             53  55  D0 001B8          MOVL    LAST_CELL, TEMP                         ; 2090
    0C  AC       18  A3                     10  00  EC 001BB          CMPV    #0, #16, 24(TEMP), SUB_CNT              ; 2092
                                                 06  13 001C2          BEQL    25$                                     ;
                                          08 A3  50  D0 001C4          MOVL    TEMP1, 8(TEMP)                          ; 2094
                                                 0A  11 001C8          BRB     27$                                     ;
                                          04 A3  50  D0 001CA 25$:     MOVL    TEMP1, @(TEMP)                          ; 2096
                                                 04  11 001CE          BRB     27$                                     ; 2087
                                          08 B7  50  D0 001D0 26$:     MOVL    TEMP1, @CELL+8                          ; 2103
                                                 52  D5 001D4 27$:     TSTL    NEXT_CELL                               ; 2108
                                                 06  13 001D6          BEQL    28$                                     ;
                                             53  52  D0 001D8          MOVL    NEXT_CELL, TEMP                         ; 2111
                                             63  50  D0 001DB          MOVL    TEMP1, (TEMP)                           ; 2112
                                             67  50  D0 001DE 28$:     MOVL    TEMP1, CELL                             ; 2118
                                                 04  001E1          RET                                             ; 2121

; Routine Size:  482 bytes,    Routine Base:  $CODE$ + 0475
```

```
;  1324          2122  1 %SBTTL 'INSERT_REF -- Insert page reference into list'
;  1325          2123  1 ROUTINE INSERT_REF (XTN) =
;  1326          2124  1 !++
;  1327          2125  1 !
;  1328          2126  1 ! FUNCTIONAL DESCRIPTION:
;  1329          2127  1 !
;  1330          2128  1 !       This routine inserts a page reference into the indexing pool
;  1331          2129  1 !
;  1332          2130  1 ! FORMAL PARAMETERS:
;  1333          2131  1 !
;  1334          2132  1 !       XTN - Transaction number
;  1335          2133  1 !
;  1336          2134  1 ! IMPLICIT INPUTS:
;  1337          2135  1 !
;  1338          2136  1 !       XPLBLK - Extended indexing attributes block
;  1339          2137  1 !       BOOKID - Master index book ident string address
;  1340          2138  1 !
;  1341          2139  1 ! IMPLICIT OUTPUTS:
;  1342          2140  1 !
;  1343          2141  1 !       None
;  1344          2142  1 !
;  1345          2143  1 ! ROUTINE VALUE:
;  1346          2144  1 ! COMPLETION CODES:
;  1347          2145  1 !
;  1348          2146  1 !       Returns the address of the saved page reference
;  1349          2147  1 !
;  1350          2148  1 ! SIDE EFFECTS:
;  1351          2149  1 !
;  1352          2150  1 !       None
;  1353          2151  1 !--
;  1354          2152  2     BEGIN
;  1355          2153  2
;  1356          2154  2     LOCAL
;  1357          2155  2         REF_CELL : $XX_BLOCK;
;  1358          2156  2
;  1359          2157  2     REF_CELL [XX$A_LINK] = 0;
;  1360          2158  2     REF_CELL [XX$A_BOOK] = .BOOKID;
;  1361          2159  2     REF_CELL [XX$H_PAGE] = .XTN;
;  1362          2160  2     REF_CELL [XX$V_FLAGS] = 0;
;  1363          2161  2     REF_CELL [XX$A_APPEND] = 0;
;  1364          2162  2
;  1365          2163  2     IF .XPLBLK [XPL$V_VALID]
;  1366          2164  2     THEN
;  1367          2165  3         BEGIN
;  1368          2166  3         !
;  1369          2167  3         ! Have .XPLUS information
;  1370          2168  3         !
;  1371          2169  3         LOCAL
;  1372          2170  3             APPEND : REF $STR_DESCRIPTOR ();
;  1373          2171  3
;  1374          2172  3         REF_CELL [XX$V_BOLD] = .XPLBLK [XPL$V_BOLD];
;  1375          2173  3         REF_CELL [XX$V_UNDERLINE] = .XPLBLK [XPL$V_UNDERLINE];
;  1376          2174  3         REF_CELL [XX$V_BEGIN] = .XPLBLK [XPL$V_BEGIN];
;  1377          2175  3         REF_CELL [XX$V_END] = .XPLBLK [XPL$V_END];
;  1378          2176  3
;  1379          2177  3         APPEND = XPLBLK [XPL$T_APPEND];
;  1380          2178  3         IF .APPEND [STR$H_LENGTH] NEQ 0
```

```
;  1381          2179  3          THEN
;  1382          2180                   !
;  1383          2181                   ! Have an append string
;  1384          2182                   !
;  1385          2183                   REF_CELL [XX$A_APPEND] = SAVDAT (.APPEND [STR$A_POINTER], DS_X_STRING, .APPEND [STR$H_LENGTH]);
;  1386          2184
;  1387          2185               END;
;  1388          2186
;  1389          2187  2        RETURN SAVDAT (REF_CELL, DS_XX_ENTRY, XX$K_LENGTH);
;  1390          2188  1        END;


                                                000C 00000 INSERT_REF:
                                                           .WORD   Save R2,R3                                          ;  2123
                                    53 00000000G   EF  9E 00002    MOVAB   SAVDAT, R3                                  ;
                                    52 00000000G   EF  9E 00009    MOVAB   XPLBLK, R2                                  ;
                                    5E             0C  C2 00010    SUBL2   #12, SP                                     ;
                                                   7E  D4 00013    CLRL    REF_CELL                                    ;  2157
                                 0C AE 00000000G   EF  D0 00015    MOVL    BOOKID, REF_CELL+12                         ;  2158
                                 08 AE       04    AC  3C 0001D    MOVZWL  XTN, REF_CELL+8                             ;  2159
                                             04    AE  D4 00022    CLRL    REF_CELL+4                                  ;  2161
                                    43             62  E9 00025    BLBC    XPLBLK, 1$                                  ;  2163
                50          62                 01  01  EF 00028    EXTZV   #1, #1, XPLBLK, R0                          ;  2172
             0A AE          01                 00  50  F0 0002D    INSV    R0, #0, #1, REF_CELL+10                     ;
                50          62                 01  02  EF 00033    EXTZV   #2, #1, XPLBLK, R0                          ;  2173
             0A AE          01                 01  50  F0 00038    INSV    R0, #1, #1, REF_CELL+10                     ;
                50          62                 01  03  EF 0003E    EXTZV   #3, #1, XPLBLK, R0                          ;  2174
             0A AE          01                 02  50  F0 00043    INSV    R0, #2, #1, REF_CELL+10                     ;
                50          62                 01  04  EF 00049    EXTZV   #4, #1, XPLBLK, R0                          ;  2175
             0A AE          01                 03  50  F0 0004E    INSV    R0, #3, #1, REF_CELL+10                     ;
                                    50       0C    A2  9E 00054    MOVAB   XPLBLK+12, APPEND                           ;  2177
                                                   60  B5 00058    TSTW    (APPEND)                                    ;  2178
                                                   0F  13 0005A    BEQL    1$                                          ;
                                    7E             60  3C 0005C    MOVZWL  (APPEND), -(SP)                             ;  2183
                                                   7E  D4 0005F    CLRL    -(SP)                                       ;
                                             04    A0  DD 00061    PUSHL   4(APPEND)                                   ;
                                    63             03  FB 00064    CALLS   #3, SAVDAT                                  ;
                                 04 AE             50  D0 00067    MOVL    R0, REF_CELL+4                              ;
                                                   04  DD 0006B 1$: PUSHL   #4                                         ;  2187
                                                   04  DD 0006D    PUSHL   #4                                          ;
                                             08    AE  9F 0006F    PUSHAB  REF_CELL                                    ;
                                    63             03  FB 00072    CALLS   #3, SAVDAT                                  ;
                                                   04  00075    RET                                                     ;  2188

; Routine Size:  118 bytes,    Routine Base:  $CODE$ + 0657
```

```
;  1392          2189  1 %SBTTL 'ENTRY_CMP -- Compare new entry with current entry'
;  1393          2190  1 ROUTINE ENTRY_CMP (NEW_PTR, NEW_LEN, LAST, XTN, LEVEL) =
;  1394          2191  1 !++
;  1395          2192  1 !
;  1396          2193  1 ! FUNCTIONAL DESCRIPTION:
;  1397          2194  1 !
;  1398          2195  1 !       This routine compares the new entry with the current entry.
;  1399          2196  1 !
;  1400          2197  1 !       For subindex entries (i.e. LEVEL NEQ 0) performs the following checks:
;  1401          2198  1 !
;  1402          2199  1 !           If LAST is TRUE, checks to see if new entry is either
;  1403          2200  1 !           a .ENTRY or .YPLUS.
;  1404          2201  1 !
;  1405          2202  1 !           If the new entry is a .Y or .YP, it checks to see if the
;  1406          2203  1 !           current entry is either a .X (.XP) or if the current entry
;  1407          2204  1 !           has subentries. If so, a value of TRUE is returned indicating
;  1408          2205  1 !           that the new entry should be inserted before the current entry.
;  1409          2206  1 !
;  1410          2207  1 !           If LAST is FALSE or the new entry is not a .Y (.YP) and the
;  1411          2208  1 !           current entry is a .Y (.YP), a value of FALSE is returned
;  1412          2209  1 !           indicating that the new entry should be inserted after the
;  1413          2210  1 !           current entry.
;  1414          2211  1 !
;  1415          2212  1 !       Otherwise, calls STRG_CMP to see if the new entry should be
;  1416          2213  1 !       inserted here. If so, calls STRG_CMP again to see if the
;  1417          2214  1 !       new entry is identical to the current entry.
;  1418          2215  1 !
;  1419          2216  1 ! FORMAL PARAMETERS:
;  1420          2217  1 !
;  1421          2218  1 !       NEW_PTR - Pointer to new entry text
;  1422          2219  1 !       NEW_LEN - Length of new entry
;  1423          2220  1 !       LAST    - TRUE if XTN should be compared to transaction number
;  1424          2221  1 !                 associated with CELL.
;  1425          2222  1 !       XTN     - Transaction number of new entry if LAST is TRUE
;  1426          2223  1 !       LEVEL   - Subindex level (0 to n)
;  1427          2224  1 !
;  1428          2225  1 ! IMPLICIT INPUTS:
;  1429          2226  1 !
;  1430          2227  1 !       CELL    - contains pointers to the list item for comparison.
;  1431          2228  1 !       SORT_LEN- Length of sort string if any
;  1432          2229  1 !       SORT_PTR- pointer to sort string if any
;  1433          2230  1 !
;  1434          2231  1 ! IMPLICIT OUTPUTS:
;  1435          2232  1 !
;  1436          2233  1 !       None
;  1437          2234  1 !
;  1438          2235  1 ! ROUTINE VALUE:
;  1439          2236  1 ! COMPLETION CODES:
;  1440          2237  1 !
;  1441          2238  1 !       TRUE if new entry should be inserted before the current entry,
;  1442          2239  1 !       FALSE otherwise.
;  1443          2240  1 !
;  1444          2241  1 ! SIDE EFFECTS:
;  1445          2242  1 !
;  1446          2243  1 !       None
;  1447          2244  1 !--
;  1448          2245  2     BEGIN
```

```
;  1449          2246  2     LOCAL
;  1450          2247  2         CEPTR : REF $XE_BLOCK,
;  1451          2248  2         N_PTR,
;  1452          2249  2         N_LEN,
;  1453          2250  2         C_VEC : REF VECTOR,
;  1454          2251  2         C_PTR,
;  1455          2252  2         C_LEN;
;  1456          2253  2
;  1457          2254  2     CEPTR = .CELL [C$A_CURR];
;  1458          2255  2
;  1459          2256  2     IF .LEVEL NEQ 0
;  1460          2257  3     THEN
;  1461          2258  3         BEGIN
;  1462          2259  3         !
;  1463          2260  3         ! Subindex entry.
;  1464          2261  3         ! Check to see if we should float a .Y
;  1465          2262  3         !
;  1466          2263  4         IF .LAST AND (.XTN EQL 0)                       ! If at bottom of new entry
;  1467          2264  3         THEN                                            ! and new entry is a .Y
;  1468          2265  4             BEGIN
;  1469          2266  5             IF (.CEPTR [XE$A_REF] NEQ 0) OR (.CEPTR [XE$A_SUBX] NEQ 0)
;  1470          2267  4             THEN
;  1471          2268  4                 !
;  1472          2269  4                 ! Current entry is a .X or .XP or has subentries.
;  1473          2270  4                 !
;  1474          2271  4                 RETURN TRUE;                        ! New entry is before current entry
;  1475          2272  4
;  1476          2273  4             END
;  1477          2274  3         ELSE
;  1478          2275  4             BEGIN
;  1479          2276  4             !
;  1480          2277  4             ! Not at bottom of entry or not .Y
;  1481          2278  4             !
;  1482          2279  5             IF (.CEPTR [XE$A_REF] EQL 0) AND (.CEPTR [XE$A_SUBX] EQL 0)
;  1483          2280  4             THEN
;  1484          2281  4                 !
;  1485          2282  4                 ! Current entry is a .Y or .YP
;  1486          2283  4                 !
;  1487          2284  4                 RETURN FALSE;                       ! New entry is after current entry
;  1488          2285  4
;  1489          2286  3             END;
;  1490          2287  2         END;
;  1491          2288  2
;  1492          2289  2     IF .SORT_LEN NEQ 0
;  1493          2290  3     THEN
;  1494          2291  3         BEGIN
;  1495          2292  3         !
;  1496          2293  3         ! New entry has a sort string. Use it.
;  1497          2294  3         !
;  1498          2295  3         N_PTR = .SORT_PTR;
;  1499          2296  3         N_LEN = .SORT_LEN;
;  1500          2297  3         END
;  1501          2298  2     ELSE
;  1502          2299  3         BEGIN
;  1503          2300  3         !
;  1504          2301  3         ! no sort string is available. use text.
;  1505          2302  3         !
```

```
;  1506          2303  3          N_PTR = .NEW_PTR;
;  1507          2304  3          N_LEN = .NEW_LEN;
;  1508          2305  3          END;
;  1509          2306  2
;  1510          2307  2      IF .CEPTR [XE$A_SORT_AS] NEQ 0
;  1511          2308  2      THEN
;  1512          2309  2          !
;  1513          2310  2          ! Current entry has a sort string. Use it.
;  1514          2311  2          !
;  1515          2312  2          C_VEC = .CEPTR [XE$A_SORT_AS]
;  1516          2313  2      ELSE
;  1517          2314  2          !
;  1518          2315  2          ! Current entry has no sort string. Use text of entry.
;  1519          2316  2          !
;  1520          2317  2          C_VEC = .CEPTR [XE$A_TEXT];
;  1521          2318  2
;  1522          2319  2      !
;  1523          2320  2      ! Get number of characters in internal sort string
;  1524          2321  2      ! Length is stored as the first fullword of the string
;  1525          2322  2      !
;  1526          2323  2      C_LEN = .C_VEC [0];
;  1527          2324  2      C_PTR = CH$PTR (C_VEC [1]);
;  1528          2325  2
;  1529          2326  2      !
;  1530          2327  2      ! Check to see if this is the proper insertion point
;  1531          2328  2      !
;  1532          2329  2      IF STRG_CMP (.N_LEN, .N_PTR, .C_LEN, .C_PTR)
;  1533          2330  2      THEN
;  1534          2331  3          BEGIN
;  1535          2332  3          !
;  1536          2333  3          ! This is almost the spot.
;  1537          2334  3          ! Check for identical sort strings.
;  1538          2335  3          !
;  1539          2336  3          IF .CELL [C$V_IDNS]
;  1540          2337  3          THEN
;  1541          2338  4              BEGIN
;  1542          2339  4              !
;  1543          2340  4              ! Sort strings were identical.
;  1544          2341  4              ! Compare text strings to determine positioning.
;  1545          2342  4              !
;  1546          2343  4              CELL [C$V_IDNS] = FALSE;
;  1547          2344  4
;  1548          2345  4              C_VEC = .CEPTR [XE$A_TEXT];
;  1549          2346  4              C_LEN = .C_VEC [0];
;  1550          2347  4              C_PTR = CH$PTR (C_VEC [1]);
;  1551          2348  4
;  1552          2349  4              RETURN STRG_CMP (.NEW_LEN, .NEW_PTR, .C_LEN, .C_PTR);
;  1553          2350  4              END
;  1554          2351  3          ELSE
;  1555          2352  3              !
;  1556          2353  3              ! Sort strings different.
;  1557          2354  3              ! This is the correct position for insertion.
;  1558          2355  3              !
;  1559          2356  3              RETURN TRUE;
;  1560          2357  3          END
;  1561          2358  2      ELSE
;  1562          2359  2          RETURN FALSE;
```

```
; 1563        2360  2
; 1564        2361  1     END;


                                    00FC 00000 ENTRY_CMP:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7                  ; 2190
                        57 00000000V   EF  9E 00002         MOVAB   STRG_CMP, R7
                        56 00000000'   EF  9E 00009         MOVAB   CELL+12, R6
                        54        F4   A6  D0 00010         MOVL    CELL, CEPTR                         ; 2254
                                  14   AC  D5 00014         TSTL    LEVEL                               ; 2256
                                       1F  13 00017         BEQL    2$
                        11        0C   AC  E9 00019         BLBC    LAST, 1$                            ; 2263
                                  10   AC  D5 0001D         TSTL    XTN
                                       0C  12 00020         BNEQ    1$
                                  0C   A4  D5 00022         TSTL    12(CEPTR)                           ; 2266
                                       66  12 00025         BNEQ    7$
                                  08   A4  D5 00027         TSTL    8(CEPTR)
                                       0C  13 0002A         BEQL    2$
                                       5F  11 0002C         BRB     7$                                  ; 2271
                                  0C   A4  D5 0002E 1$:     TSTL    12(CEPTR)                           ; 2279
                                       05  12 00031         BNEQ    2$
                                  08   A4  D5 00033         TSTL    8(CEPTR)
                                       59  13 00036         BEQL    8$
                        50      04B8   C6  D0 00038 2$:     MOVL    SORT_LEN, R0                        ; 2289
                                       07  13 0003D         BEQL    3$
                        51      04B4   C6  D0 0003F         MOVL    SORT_PTR, N_PTR                     ; 2295
                                       08  11 00044         BRB     4$                                  ; 2289
                        51        04   AC  D0 00046 3$:     MOVL    NEW_PTR, N_PTR                      ; 2303
                        50        08   AC  D0 0004A         MOVL    NEW_LEN, N_LEN                      ; 2304
                                  14   A4  D5 0004E 4$:     TSTL    20(CEPTR)                           ; 2307
                                       06  13 00051         BEQL    5$
                        52        14   A4  D0 00053         MOVL    20(CEPTR), C_VEC                    ; 2312
                                       04  11 00057         BRB     6$
                        52        10   A4  D0 00059 5$:     MOVL    16(CEPTR), C_VEC                    ; 2317
                        55             62  D0 0005D 6$:     MOVL    (C_VEC), C_LEN                      ; 2323
                        53        04   A2  9E 00060         MOVAB   4(R2), C_PTR                        ; 2324
                                       53  DD 00064         PUSHL   C_PTR                               ; 2329
                                       23  BB 00066         PUSHR   #^M<R0,R1,R5>
                        67             04  FB 00068         CALLS   #4, STRG_CMP
                        23             50  E9 0006B         BLBC    R0, 8$
                        1C             66  E9 0006E         BLBC    CELL+12, 7$                         ; 2336
                        66             01  8A 00071         BICB2   #1, CELL+12                         ; 2343
                        52        10   A4  D0 00074         MOVL    16(CEPTR), C_VEC                    ; 2345
                        55             62  D0 00078         MOVL    (C_VEC), C_LEN                      ; 2346
                        53        04   A2  9E 0007B         MOVAB   4(R2), C_PTR                        ; 2347
                                       53  DD 0007F         PUSHL   C_PTR                               ; 2349
                                       55  DD 00081         PUSHL   C_LEN
                                  04   AC  DD 00083         PUSHL   NEW_PTR
                                  08   AC  DD 00086         PUSHL   NEW_LEN
                        67             04  FB 00089         CALLS   #4, STRG_CMP
                                       04  0008C         RET                                         ; 2356
                        50             01  D0 0008D 7$:     MOVL    #1, R0
                                       04  00090         RET                                         ; 2359
                                       50  D4 00091 8$:     CLRL    R0                                  ; 2361
```

```
                                                  04 00093        RET                                                     ;

; Routine Size:  148 bytes,    Routine Base:  $CODE$ + 06CD
```

```
;  1566          2362  1 %SBTTL 'STRG_CMP -- Compare two strings'
;  1567          2363  1
;  1568          2364  1 ROUTINE STRG_CMP (S1_LEN, S1_PTR, S2_LEN, S2_PTR) =
;  1569          2365  1
;  1570          2366  1 !++
;  1571          2367  1 ! FUNCTIONAL DESCRIPTION:
;  1572          2368  1 !
;  1573          2369  1 !       This routine is called to compare two strings.
;  1574          2370  1 !       It returns TRUE if string 1 should be before string 2.
;  1575          2371  1 !
;  1576          2372  1 !       It sets CELL [C$V_IDNS] if the strings are identical.
;  1577          2373  1 !
;  1578          2374  1 ! FORMAL PARAMETERS:
;  1579          2375  1 !
;  1580          2376  1 !       S1_LEN - Length of string 1
;  1581          2377  1 !       S1_PTR - Pointer to string 1
;  1582          2378  1 !       S2_LEN - Length of string 2
;  1583          2379  1 !       S2_PTR - Pointer to string 2
;  1584          2380  1 !
;  1585          2381  1 ! IMPLICIT INPUTS:
;  1586          2382  1 !
;  1587          2383  1 !       NONE
;  1588          2384  1 !
;  1589          2385  1 ! IMPLICIT OUTPUTS:
;  1590          2386  1 !
;  1591          2387  1 !       CELL [C$V_IDNS] - set to true if strings are identical
;  1592          2388  1 !
;  1593          2389  1 ! ROUTINE VALUE:
;  1594          2390  1 ! COMPLETION CODES:
;  1595          2391  1 !
;  1596          2392  1 !       TRUE    - String 1 is before string 2
;  1597          2393  1 !       FALSE   - Otherwise
;  1598          2394  1 !
;  1599          2395  1 ! SIDE EFFECTS:
;  1600          2396  1 !
;  1601          2397  1 !       NONE
;  1602          2398  1 !
;  1603          2399  1 !--
;  1604          2400  1
;  1605          2401  2     BEGIN
;  1606          2402  2
;  1607          2403  2     LOCAL
;  1608          2404  2         CASECMP,
;  1609          2405  2         CHARCMP,
;  1610          2406  2         COLUMN,
;  1611          2407  2         EMPHCMP,
;  1612          2408  2         ICASE,
;  1613          2409  2         IEMPH,
;  1614          2410  2         OLDCASE,
;  1615          2411  2         OLDEMPH,
;  1616          2412  2         PTR_1,
;  1617          2413  2         REM_1,
;  1618          2414  2         PTR_2,
;  1619          2415  2         REM_2;
;  1620          2416  2
;  1621          2417  2     PTR_1 = .S1_PTR;
;  1622          2418  2     REM_1 = .S1_LEN;
```

```
;  1623     2419  2      PTR_2 = .S2_PTR;
;  1624     2420  2      REM_2 = .S2_LEN;
;  1625     2421  2
;  1626     2422  2      ICASE = 0;                              ! No differences in case yet
;  1627     2423  2      IEMPH = 0;                              ! No differences in emphasis yet
;  1628     2424  2      OLDCASE = 0;                            ! No differences in case yet
;  1629     2425  2      OLDEMPH = 0;                            ! No differences in emphasis yet
;  1630     2426  2      COLUMN = 0;                             ! No print positions scanned yet.
;  1631     2427  2
;  1632     2428  2      !
;  1633     2429  2      ! Loop until done with both strings
;  1634     2430  2      !
;  1635     2431  2      REPEAT
;  1636     2432  3          BEGIN
;  1637     2433  3          !
;  1638     2434  3          ! Update count of print columns, so positions of case and emphasis
;  1639     2435  3          ! differences can be remembered.
;  1640     2436  3          !
;  1641     2437  3          COLUMN = .COLUMN + 1;
;  1642     2438  3
;  1643     2439  3          !
;  1644     2440  3          ! Make sure neither string has run out.  If one has, this is
;  1645     2441  3          ! the place for insertion.
;  1646     2442  3          !
;  1647     2443  4          IF (.REM_2 LEQ 0) OR (.REM_1 LEQ 0)
;  1648     2444  3          THEN
;  1649     2445  4              BEGIN
;  1650     2446  4              !
;  1651     2447  4              ! Check for exact string before leaving
;  1652     2448  4              !
;  1653     2449  5              IF (.REM_2 LEQ 0) AND (.REM_1 LEQ 0)
;  1654     2450  4              THEN
;  1655     2451  4                  !
;  1656     2452  4                  ! Both strings have run out.  They're identical if
;  1657     2453  4                  ! there are no case or emphasis differences.
;  1658     2454  4                  !
;  1659     2455  5                  CELL [C$V_IDNS] = ((.ICASE EQL 0) AND (.IEMPH EQL 0))
;  1660     2456  4              ELSE
;  1661     2457  4                  !
;  1662     2458  4                  ! Only one string has run out.  The longer of the two strings
;  1663     2459  4                  ! is "greater" than the shorter string, or conversely, the
;  1664     2460  4                  ! one that's run out is the "lesser" of the two.  Return TRUE
;  1665     2461  4                  ! if the input string is the "lesser" of the two.
;  1666     2462  4                  !
;  1667     2463  4                  RETURN (.REM_1 LEQ 0);
;  1668     2464  4
;  1669     2465  4              IF .OLDEMPH NEQ 0 THEN RETURN (.OLDEMPH EQL 1);
;  1670     2466  4
;  1671     2467  4              IF .OLDCASE NEQ 0 THEN RETURN (.OLDCASE EQL -1);
;  1672     2468  4
;  1673     2469  4              RETURN TRUE;
;  1674     2470  3              END;
;  1675     2471  3
;  1676     2472  3          CHRCMP (PTR_1, PTR_2, CASECMP, CHARCMP, EMPHCMP, REM_1, REM_2);
;  1677     2473  3
;  1678     2474  3          IF .CHARCMP NEQ 0
;  1679     2475  3          THEN
```

```
;  1680     2476  4            RETURN (.CHARCMP EQL -1)
;  1681     2477  3        ELSE
;  1682     2478  4            BEGIN
;  1683     2479  4            !
;  1684     2480  4            ! Remember differences in the string so they can be
;  1685     2481  4            ! applied if the string runs out.
;  1686     2482  4            !
;  1687     2483  4            ! If there is a difference of cases, the very first place where
;  1688     2484  4            ! case differs is the significant case difference.  All other
;  1689     2485  4            ! positions are secondary.
;  1690     2486  4            !
;  1691     2487  5            IF (.ICASE EQL 0) AND (.CASECMP NEQ 0)
;  1692     2488  4            THEN
;  1693     2489  5                BEGIN
;  1694     2490  5                !
;  1695     2491  5                ! Remember column position where difference occurred
;  1696     2492  5                ! Remember what the case difference was.
;  1697     2493  5                !
;  1698     2494  5                ICASE = .COLUMN;
;  1699     2495  5                OLDCASE = .CASECMP;
;  1700     2496  4                END;
;  1701     2497  4
;  1702     2498  4            !
;  1703     2499  4            ! If there is a difference in emphasis, the very first place where
;  1704     2500  4            ! emphasis differs is the significant emphasis difference.
;  1705     2501  4            ! All other positions are secondary.
;  1706     2502  4            !
;  1707     2503  5            IF (.IEMPH EQL 0) AND (.EMPHCMP NEQ 0)
;  1708     2504  4            THEN
;  1709     2505  5                BEGIN
;  1710     2506  5                !
;  1711     2507  5                ! Remember column position where difference occurred.
;  1712     2508  5                ! Remember what the difference in emphasis was.
;  1713     2509  5                !
;  1714     2510  5                IEMPH = .COLUMN;
;  1715     2511  5                OLDEMPH = .EMPHCMP;
;  1716     2512  4                END;
;  1717     2513  3            END;
;  1718     2514  3
;  1719     2515  3        END
;  1720     2516  3
;  1721     2517  1    END;                        !End of STRG_CMP


                                00FC 00000 STRG_CMP:
                                                .WORD   Save R2,R3,R4,R5,R6,R7          ; 2364
                          5E    14   C2 00002           SUBL2   #20, SP                 ;
                    10    AE    08   AC  D0 00005       MOVL    S1_PTR, PTR_1           ; 2417
                                04   AC  DD 0000A       PUSHL   S1_LEN                  ; 2418
                    10    AE    10   AC  D0 0000D       MOVL    S2_PTR, PTR_2           ; 2419
                                0C   AC  DD 00012       PUSHL   S2_LEN                  ; 2420
                                     54  7C 00015       CLRQ    IEMPH                   ; 2423
                                     52  7C 00017       CLRQ    OLDEMPH                 ; 2425
                                     56  D4 00019       CLRL    COLUMN                  ; 2426
```

```
                                                 56  D6 0001B 1$:    INCL    COLUMN                          ; 2437
                                                 50  D4 0001D        CLRL    R0                              ; 2443
                                                 6E  D5 0001F        TSTL    REM_2
                                                 04  14 00021        BGTR    2$
                                                 50  D6 00023        INCL    R0
                                                 05  11 00025        BRB     3$
                                             04  AE  D5 00027 2$:    TSTL    REM_1
                                                 50  14 0002A        BGTR    10$
                                         27      50  E9 0002C 3$:    BLBC    R0, 6$                          ; 2449
                                             04  AE  D5 0002F        TSTL    REM_1
                                                 22  14 00032        BGTR    6$
                                                 51  D4 00034        CLRL    R1                              ; 2455
                                                 55  D5 00036        TSTL    ICASE
                                                 02  12 00038        BNEQ    4$
                                                 51  D6 0003A        INCL    R1
                                                 50  D4 0003C 4$:    CLRL    R0
                                                 54  D5 0003E        TSTL    IEMPH
                                                 02  12 00040        BNEQ    5$
                                                 50  D6 00042        INCL    R0
                                         57      51  D2 00044 5$:    MCOML   R1, R7
                                 57      50  8B  57  8B 00047        BICB3   R7, R0, R7
00000000'  EF                    01      00  F0  57  F0 0004B        INSV    R7, #0, #1, CELL+12
                                                 08  11 00054        BRB     7$
                                                 50  D4 00056 6$:    CLRL    R0                              ; 2463
                                             04  AE  D5 00058        TSTL    REM_1
                                                 4B  15 0005B        BLEQ    12$
                                                     04 0005D        RET
                                                 52  D5 0005E 7$:    TSTL    OLDEMPH                         ; 2465
                                                 07  13 00060        BEQL    8$
                                                 50  D4 00062        CLRL    R0
                                         01      52  D1 00064        CMPL    OLDEMPH, #1
                                                 3D  11 00067        BRB     11$
                                                 53  D5 00069 8$:    TSTL    OLDCASE                         ; 2467
                                                 0B  13 0006B        BEQL    9$
                                                 50  D4 0006D        CLRL    R0
                                 FFFFFFFF    BF  53  D1 0006F        CMPL    OLDCASE, #-1
                                                 2E  11 00076        BRB     11$
                                         50      01  D0 00078 9$:    MOVL    #1, R0                          ; 2469
                                                     04 0007B        RET
                                                 5E  DD 0007C 10$:    PUSHL   SP                              ; 2472
                                             08  AE  9F 0007E        PUSHAB  REM_1
                                             10  AE  9F 00081        PUSHAB  EMPHCMP
                                             18  AE  9F 00084        PUSHAB  CHARCMP
                                             20  AE  9F 00087        PUSHAB  CASECMP
                                             28  AE  9F 0008A        PUSHAB  PTR_2
                                             30  AE  9F 0008D        PUSHAB  PTR_1
                                 00000000V   EF  07  FB 00090        CALLS   #7, CHRCMP
                                             0C  AE  D5 00097        TSTL    CHARCMP                         ; 2474
                                                 0F  13 0009A        BEQL    13$
                                                 50  D4 0009C        CLRL    R0                              ; 2476
                                 FFFFFFFF    BF  0C  AE  D1 0009E        CMPL    CHARCMP, #-1
                                                 26  12 000A6 11$:    BNEQ    16$
                                                 50  D6 000AB 12$:    INCL    R0
                                                     04 000AA        RET
                                                 55  D5 000AB 13$:    TSTL    ICASE                           ; 2487
                                                 0C  12 000AD        BNEQ    14$
                                             10  AE  D5 000AF        TSTL    CASECMP
```

```
                                07  13 000B2          BEQL    14$                                     ;
                             55 56  D0 000B4          MOVL    COLUMN, ICASE                           ; 2494
                             53 10  AE D0 000B7          MOVL    CASECMP, OLDCASE                    ; 2495
                                54  D5 000BB 14$:       TSTL    IEMPH                               ; 2503
                                0C  12 000BD          BNEQ    15$                                     ;
                             08 AE  D5 000BF          TSTL    EMPHCMP                                 ;
                                07  13 000C2          BEQL    15$                                     ;
                             54 56  D0 000C4          MOVL    COLUMN, IEMPH                           ; 2510
                             52 08  AE D0 000C7          MOVL    EMPHCMP, OLDEMPH                    ; 2511
                          FF4D      31 000CB 15$:       BRW     1$                                  ; 2426
                                04  00CE 16$:          RET                                         ; 2517

; Routine Size:  207 bytes,    Routine Base:  $CODE$ + 0761
```

```
;  1723     2518  1 %SBTTL 'CHRCMP -- Compare two characters in internal format'
;  1724     2519  1
;  1725     2520  1 ROUTINE CHRCMP (XA, XB, CASECMP, CHARCMP, EMPHCMP, REMAINDER_A, REMAINDER_B) : NOVALUE =
;  1726     2521  1
;  1727     2522  1 !++
;  1728     2523  1 ! FUNCTIONAL DESCRIPTION:
;  1729     2524  1 !
;  1730     2525  1 !       CHRCMP compares two characters in RUNOFF internal format (i.e,
;  1731     2526  1 !       as generated by SCANT).
;  1732     2527  1 !
;  1733     2528  1 !       Basically, the comparison is done lexically, with the change
;  1734     2529  1 !       that the characters which are not letters are lexically smaller
;  1735     2530  1 !       than any letters.
;  1736     2531  1 !
;  1737     2532  1 !       It takes overstriking, underlining, and bolding into account
;  1738     2533  1 !       when doing the comparison. If two characters are identical
;  1739     2534  1 !       except for their emphasis, the comparison is such that the
;  1740     2535  1 !       character with the most emphasis is lexically smallest.
;  1741     2536  1 !
;  1742     2537  1 !       Bolding is considered to emphasize more than underlining, but
;  1743     2538  1 !       less than both underlining and bolding together. Underlining
;  1744     2539  1 !       emphasizes more than overstriking; but note that the
;  1745     2540  1 !       overstriking sequence is NOT taken into account in the
;  1746     2541  1 !       comparison. Upper case emphasizes more than lower case. Emphasis
;  1747     2542  1 !       is always less significant than "naked character" differences.
;  1748     2543  1 !
;  1749     2544  1 ! FORMAL PARAMETERS:
;  1750     2545  1 !
;  1751     2546  1 !       XA and XB are CH$PTRs to the characters to be compared.
;  1752     2547  1 !
;  1753     2548  1 !       CHARCMP - Returned as if computed by subtracting the internal
;  1754     2549  1 !                 representations of the characters, except that letters
;  1755     2550  1 !                 are "greater" than all other characters.
;  1756     2551  1 !
;  1757     2552  1 !       CASECMP - Returned as if computed by subtracting the
;  1758     2553  1 !                 "upper/lower caseness" of the characters.  Upper case
;  1759     2554  1 !                 has "value" 0, lower case 1.
;  1760     2555  1 !                 By definition, characters other than letters are in upper case.
;  1761     2556  1 !
;  1762     2557  1 !       EMPHCMP - Returned as SIGN(emphasis of A - emphasis of B) where
;  1763     2558  1 !                 each emphasis type requires one bit.  Overstriking has
;  1764     2559  1 !                 the value 1, underlining has the value 2 and bolding
;  1765     2560  1 !                 the value 4.
;  1766     2561  1 !
;  1767     2562  1 !       REMAINDER_A - the number of characters scanned in XA is subtracted from it.
;  1768     2563  1 !       REMAINDER_B - the number of characters scanned in XB in subtracted from it.
;  1769     2564  1 !
;  1770     2565  1 ! IMPLICIT INPUTS:
;  1771     2566  1 !
;  1772     2567  1 !       The arrangement of the internal representation is implicit
;  1773     2568  1 !       in the algorithm.  Basically it assumes that the "naked"
;  1774     2569  1 !       character comes after the escape (=emphasis) sequences.
;  1775     2570  1 !
;  1776     2571  1 ! IMPLICIT OUTPUTS:
;  1777     2572  1 !
;  1778     2573  1 !       NONE
;  1779     2574  1 !
```

```
;  1780          2575  1 ! ROUTINE VALUE:
;  1781          2576  1 !
;  1782          2577  1 !       The result is returned as if it could be computed by
;  1783          2578  1 !       SIGN (.A - .B);
;  1784          2579  1 !
;  1785          2580  1 ! SIDE EFFECTS:
;  1786          2581  1 !
;  1787          2582  1 !       NONE
;  1788          2583  1 !
;  1789          2584  1 !--
;  1790          2585  1
;  1791          2586  2     BEGIN
;  1792          2587  2
;  1793          2588  2     BIND
;  1794          2589  2         PTR_A = .XA,
;  1795          2590  2         PTR_B = .XB;
;  1796          2591  2
;  1797          2592  2     LOCAL
;  1798          2593  2         CA,
;  1799          2594  2         CB,
;  1800          2595  2         RA,
;  1801          2596  2         RB;
;  1802          2597  2
;  1803          2598  2     RA = 0;                             ! Assume no emphasis for character A
;  1804          2599  2     RB = 0;                             ! Assume no emphasis for character B
;  1805          2600  2     .CASECMP = 0;                       ! Assume no case difference.
;  1806          2601  2     .CHARCMP = 0;                       ! Assume characters identical.
;  1807          2602  2     .EMPHCMP = 0;                       ! Assume no emphasis at all.
;  1808          2603  2
;  1809          2604  2     REPEAT
;  1810          2605  3         BEGIN
;  1811          2606  3         CA = CH$RCHAR_A (PTR_A);
;  1812          2607  3         .REMAINDER_A = ..REMAINDER_A - 1;       ! Subtract off scanned character
;  1813          2608  3
;  1814          2609  3         IF .CA EQL RINTES
;  1815          2610  3         THEN
;  1816          2611  4             BEGIN
;  1817          2612  4             !
;  1818          2613  4             ! Interpret escape sequence.
;  1819          2614  4             !
;  1820          2615  4             CA = CH$RCHAR_A (PTR_A);
;  1821          2616  4             .REMAINDER_A = ..REMAINDER_A - 2;   ! Subtract off scanned characters.
;  1822          2617  4
;  1823          2618  4             SELECTONE .CA OF
;  1824          2619  4             SET
;  1825          2620  4
;  1826          2621  4             [%C'B']:
;  1827          2622  4                 !
;  1828          2623  4                 ! Emphasis value of bolding.
;  1829          2624  4                 !
;  1830          2625  4                 RA = .RA OR 4;
;  1831          2626  4
;  1832          2627  4             [%C'U']:
;  1833          2628  4                 !
;  1834          2629  4                 ! Emphasis value for underlining.
;  1835          2630  4                 !
;  1836          2631  4                 RA = .RA OR 2;
```

```
;  1837     2632  4
;  1838     2633  4                [%C'C']:
;  1839     2634  4                    !
;  1840     2635  4                    ! Emphasis value for overstriking.
;  1841     2636  4                    !
;  1842     2637  4                    RA = .RA OR 1;
;  1843     2638  4
;  1844     2639  4                [OTHERWISE]:
;  1845     2640  4                    !
;  1846     2641  4                    ! Non-emphasis value (do nothing)
;  1847     2642  4                    !
;  1848     2643  4                    ;
;  1849     2644  4
;  1850     2645  4                TES;
;  1851     2646  4
;  1852     2647  4                CH$RCHAR_A (PTR_A);
;  1853     2648  4                END
;  1854     2649  3            ELSE
;  1855     2650  4                BEGIN
;  1856     2651  4
;  1857     2652  5                IF UPPER_LETTER (.CA)
;  1858     2653  4                THEN
;  1859     2654  5                    CA = LOWER_CASE (.CA)
;  1860     2655  4                ELSE
;  1861     2656  4                    .CASECMP = 1;
;  1862     2657  4                EXITLOOP
;  1863     2658  4                END
;  1864     2659  2            END;
;  1865     2660  2
;  1866     2661  2            !
;  1867     2662  2            ! Scan second character.
;  1868     2663  2            !
;  1869     2664  2            REPEAT
;  1870     2665  3                BEGIN
;  1871     2666  3                CB = CH$RCHAR_A (PTR_B);
;  1872     2667  3                .REMAINDER_B = ..REMAINDER_B - 1;       ! Subtract off scanned character
;  1873     2668  3
;  1874     2669  3                IF .CB EQL RINTES
;  1875     2670  3                THEN
;  1876     2671  4                    BEGIN
;  1877     2672  4                    !
;  1878     2673  4                    ! Interpret escape sequence.
;  1879     2674  4                    !
;  1880     2675  4                    CB = CH$RCHAR_A (PTR_B);
;  1881     2676  4                    .REMAINDER_B = ..REMAINDER_B - 2;   ! Subtract off scanned characters
;  1882     2677  4                    SELECTONE .CB OF
;  1883     2678  4                    SET
;  1884     2679  4
;  1885     2680  4                    [%C'B']:
;  1886     2681  4                        !
;  1887     2682  4                        ! Emphasis value for bolding.
;  1888     2683  4                        !
;  1889     2684  4                        RB = .RB OR 4;
;  1890     2685  4
;  1891     2686  4                    [%C'U']:
;  1892     2687  4                        !
;  1893     2688  4                        ! Emphasis value for underlining
```

```
;  1894     2689  4                    !
;  1895     2690  4                    RB = .RB OR 2;
;  1896     2691  4
;  1897     2692  4                [%C'O']:
;  1898     2693  4                    !
;  1899     2694  4                    ! Emphasis value for overstriking
;  1900     2695  4                    !
;  1901     2696  4                    RB = .RB OR 1;
;  1902     2697  4
;  1903     2698  4                [OTHERWISE]:
;  1904     2699  4                    !
;  1905     2700  4                    ! Non-emphasis value (do nothing)
;  1906     2701  4                    !
;  1907     2702  4                    ;
;  1908     2703  4
;  1909     2704  4                TES;
;  1910     2705  4
;  1911     2706  4                CH$RCHAR_A (PTR_B);
;  1912     2707  4                END
;  1913     2708  3            ELSE
;  1914     2709  4                BEGIN
;  1915     2710  4
;  1916     2711  5                IF UPPER_LETTER (.CB)
;  1917     2712  4                THEN
;  1918     2713  5                    CB = LOWER_CASE (.CB)
;  1919     2714  4                ELSE
;  1920     2715  4                    .CASECMP = ..CASECMP - 1;
;  1921     2716  4                EXITLOOP
;  1922     2717  4                END
;  1923     2718  2            END;
;  1924     2719  2
;  1925     2720  2        !
;  1926     2721  2        ! At this point, the "naked" characters are in CA and CB.
;  1927     2722  2        ! Decoded emphasis escape sequences are in RA and RB.
;  1928     2723  2        !
;  1929     2724  2        ! "Subtract" emphasis to get relationship.
;  1930     2725  2        !
;  1931     2726  2        .EMPHCMP = SIGN (.RA - .RB);
;  1932     2727  2
;  1933     2728  2        !
;  1934     2729  2        ! Compare the "naked" part of the characters and
;  1935     2730  2        ! return the relationship.
;  1936     2731  2        !
;  1937     2732  3        IF LOWER_LETTER (.CA)
;  1938     2733  2        THEN                            ! First character is lower case
;  1939     2734  3            IF LOWER_LETTER (.CB)
;  1940     2735  2            THEN                        ! Second character is lower case
;  1941     2736  2                .CHARCMP = SIGN (.CA - .CB) ! return relationship between characters
;  1942     2737  2            ELSE                        ! Second character is upper case
;  1943     2738  2                .CHARCMP = 1            ! hence second character is "largest".
;  1944     2739  2
;  1945     2740  2        ELSE                            ! First character is upper case
;  1946     2741  3            IF LOWER_LETTER (.CB)
;  1947     2742  2            THEN                        ! Second character is lower case
;  1948     2743  2                .CHARCMP = -1           ! hence first character is "largest"
;  1949     2744  2            ELSE                        ! Second character is upper case
;  1950     2745  2                .CHARCMP = SIGN (.CA - .CB);! return relationship between characters
```

```
;  1951        2746  2
;  1952        2747  1     END;                          !End of CHRCMP


                                    007C 00000 CHRCMP:  .WORD   Save R2,R3,R4,R5,R6                 ; 2520
                          56    00G 8F  9A 00002        MOVZBL  #RINTES, R6                       ;
                                    54  D4 00006        CLRL    RA                                ; 2598
                                    51  D4 00008        CLRL    RB                                ; 2599
                                0C  BC  D4 0000A        CLRL    @CASECMP                          ; 2600
                          53    10  AC  D0 0000D        MOVL    CHARCMP, R3                       ; 2601
                                    63  D4 00011        CLRL    (R3)                              ;
                                14  BC  D4 00013        CLRL    @EMPHCMP                          ; 2602
                          50    04  BC  D0 00016 1$:    MOVL    @XA, R0                           ; 2606
                          52        60  9A 0001A        MOVZBL  (R0), CA                          ;
                                04  BC  D6 0001D        INCL    @XA                               ;
                                18  BC  D7 00020        DECL    @REMAINDER_A                      ; 2607
                          56            52  D1 00023    CMPL    CA, R6                            ; 2609
                                        3B  12 00026    BNEQ    5$                                ;
                          50    04  BC  D0 00028        MOVL    @XA, R0                           ; 2615
                          52        60  9A 0002C        MOVZBL  (R0), CA                          ;
                                04  BC  D6 0002F        INCL    @XA                               ;
                      18  BC            02  C2 00032    SUBL2   #2, @REMAINDER_A                  ; 2616
              00000042    8F            52  D1 00036    CMPL    CA, #66                           ; 2621
                                        05  12 0003D    BNEQ    2$                                ;
                          54            04  88 0003F    BISB2   #4, RA                            ; 2625
                                        1A  11 00042    BRB     4$                                ;
              00000055    8F            52  D1 00044 2$:    CMPL    CA, #85                       ; 2627
                                        05  12 0004B    BNEQ    3$                                ;
                          54            02  88 0004D    BISB2   #2, RA                            ; 2631
                                        0C  11 00050    BRB     4$                                ;
              0000004F    8F            52  D1 00052 3$:    CMPL    CA, #79                       ; 2633
                                        03  12 00059    BNEQ    4$                                ;
                          54            01  88 0005B    BISB2   #1, RA                            ; 2637
                                04  BC  D6 0005E 4$:    INCL    @XA                               ; 2647
                                        B3  11 00061    BRB     1$                                ; 2609
              00000041    8F            52  D1 00063 5$:    CMPL    CA, #65                       ; 2652
                                        0E  19 0006A    BLSS    6$                                ;
              0000005A    8F            52  D1 0006C        CMPL    CA, #90                       ;
                                        05  14 00073    BGTR    6$                                ;
                          52            20  C0 00075    ADDL2   #32, CA                           ; 2654
                                        04  11 00078    BRB     7$                                ;
                      0C  BC            01  D0 0007A 6$:    MOVL    #1, @CASECMP                  ; 2656
                          55    08  BC  D0 0007E 7$:    MOVL    @XB, R5                           ; 2666
                          50        65  9A 00082        MOVZBL  (R5), CB                          ;
                                08  BC  D6 00085        INCL    @XB                               ;
                                1C  BC  D7 00088        DECL    @REMAINDER_B                      ; 2667
                          56            50  D1 0008B    CMPL    CB, R6                            ; 2669
                                        3B  12 0008E    BNEQ    11$                               ;
                          55    08  BC  D0 00090        MOVL    @XB, R5                           ; 2675
                          50        65  9A 00094        MOVZBL  (R5), CB                          ;
                                08  BC  D6 00097        INCL    @XB                               ;
                      1C  BC            02  C2 0009A    SUBL2   #2, @REMAINDER_B                  ; 2676
              00000042    8F            50  D1 0009E    CMPL    CB, #66                           ; 2680
                                        05  12 000A5    BNEQ    8$
```

```
                                 51              04  88 000A7          BISB2   #4, RB                          ; 2684
                                                 1A  11 000AA          BRB     10$                             ;
                          00000055   8F          50  D1 000AC 8$:      CMPL    CB, #85                         ; 2686
                                                 05  12 000B3          BNEQ    9$                              ;
                                 51              02  88 000B5          BISB2   #2, RB                          ; 2690
                                                 0C  11 000B8          BRB     10$                             ;
                          0000004F   8F          50  D1 000BA 9$:      CMPL    CB, #79                         ; 2692
                                                 03  12 000C1          BNEQ    10$                             ;
                                 51              01  88 000C3          BISB2   #1, RB                          ; 2696
                                          08     BC  D6 000C6 10$:     INCL    @XB                             ; 2706
                                                 B3  11 000C9          BRB     7$                              ; 2669
                          00000041   8F          50  D1 000CB 11$:     CMPL    CB, #65                         ; 2711
                                                 0E  19 000D2          BLSS    12$                             ;
                          0000005A   8F          50  D1 000D4          CMPL    CB, #90                         ;
                                                 05  14 000DB          BGTR    12$                             ;
                                 50              20  C0 000DD          ADDL2   #32, CB                         ; 2713
                                                 03  11 000E0          BRB     13$                             ;
                                          0C     BC  D7 000E2 12$:     DECL    @CASECMP                        ; 2715
                                 51              54  C2 000E5 13$:     SUBL2   RA, R1                          ; 2726
                                                 51  D5 000E8          TSTL    R1                              ;
                                                 51  DC 000EA          MOVPSL  R1                              ;
                51          51                   02  02 EF 000EC          EXTZV   #2, #2, R1, R1                  ;
                                              14 BC  FF A1 9E 000F1          MOVAB   -1(R1), @EMPHCMP                ;
                          00000061   8F          52  D1 000F6          CMPL    CA, #97                         ; 2732
                                                 1F  19 000FD          BLSS    15$                             ;
                          0000007A   8F          52  D1 000FF          CMPL    CA, #122                        ;
                                                 16  14 00106          BGTR    15$                             ;
                          00000061   8F          50  D1 00108          CMPL    CB, #97                         ; 2734
                                                 09  19 0010F          BLSS    14$                             ;
                          0000007A   8F          50  D1 00111          CMPL    CB, #122                        ;
                                                 1A  15 00118          BLEQ    16$                             ;
                                 63              01  D0 0011A 14$:     MOVL    #1, (R3)                        ; 2738
                                                 04  00011D          RET                                     ; 2734
                          00000061   8F          50  D1 0011E 15$:     CMPL    CB, #97                         ; 2741
                                                 0D  19 00125          BLSS    16$                             ;
                          0000007A   8F          50  D1 00127          CMPL    CB, #122                        ;
                                                 04  14 0012E          BGTR    16$                             ;
                                 63              01  CE 00130          MNEGL   #1, (R3)                        ; 2743
                                                 04  00133          RET                                     ;
                                 50              52  C2 00134 16$:     SUBL2   CA, R0                          ; 2745
                                                 50  D5 00137          TSTL    R0                              ;
                                                 50  DC 00139          MOVPSL  R0                              ;
          50          50                         02  02 EF 0013B          EXTZV   #2, #2, R0, R0                  ;
                                                 63     FF A0 9E 00140          MOVAB   -1(R0), (R3)            ;
                                                        04 00144          RET                                     ; 2747

; Routine Size:  325 bytes,    Routine Base:  $CODE$ + 0830

; 1953          2748  1 END                                 !End of module
; 1954          2749  0 ELUDOM


                                                                    .EXTRN  LIB$SIGNAL
```

```
;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  $OWN$                         1232  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                        2421  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $PLIT$                           4  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;
;
;                                 Library Statistics
;                                          -------- Symbols --------      Pages       Processing
;       File                               Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]XPORT.L32;1           590        42         7         252        00:00.1
;
;
;
;                                   COMMAND QUALIFIERS
;
;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:NDXOUT/OBJ=OBJ$:NDXOUT MSRC$:NDXOUT/UPDATE=(ENH$:NDXOUT)
;
; Size:         2421 code + 1236 data bytes
; Run Time:       00:59.8
; Elapsed Time:   01:59.5
; Lines/CPU Min:   2760
; Lexemes/CPU-Min: 38622
; Memory Used:  232 pages
; Compilation Complete
```

NDXMSG
LIS

NDXOUT
LIS

NDXINI
LIS

NDXTEX
LIS

NDXTMS
LIS

NDXPAG
LIS

NDXVMS
LIS

NDXVMSMSG
LIS